Private Cloud Solution Package for OpenStack

Deploy and validate OpenStack Cloud with F5 LBaaS on Red Hat OpenStack Platform version 9

Mark Dittmer, F5 Networks | **John Gruber**, F5 Networks | **Paul Pindell**, F5 Networks | **Dave Cain**, Red Hat

Table of Contents

INTRODUCTION **4**
OpenStack for Cloud 4
Planning an OpenStack Architecture 4
Seamless Migration to the Cloud 4
F5 Private Cloud Solution Package for OpenStack 5
Testing the solution 5
Use Case: Migrating Workloads to OpenStack Private Cloud 5

ABOUT RED HAT OPENSTACK PLATFORM **7**

COMPONENTS **8**
1.1. Networking 10
1.1.1. OpenStack Networking (neutron) *10*
1.2. Storage 12
Section 1.2.1. "OpenStack Block Storage (cinder)" *12*
Section 1.2.2. "OpenStack Object Storage (swift)" *13*
1.3 Virtual Machines, Images and Templates 13
1.3.1. OpenStack Compute (nova) *13*
1.3.2. OpenStack Bare Metal Provisioning (ironic) *14*
1.3.3. OpenStack Image (glance) *14*
1.3.4. OpenStack Orchestration (heat) *14*
1.4. Identity Management 15
1.4.1. OpenStack Identity (keystone) *15*
1.5. User Interfaces 16
1.5.1. OpenStack Dashboard (horizon) *16*
1.6. F5 BIG-IP Platform 17
1.6.1. The Advantages of F5 BIG-IP i500 Hardware *17*
1.6.2. BIG-IP Virtual Editions *19*
1.6.3. BIG-IP TMOS Specifications *19*
1.6.4. Centralized Management and Licensing with BIG-IQ *19*
1.6.5. F5 Virtual Edition Software Modules *19*

DEPLOYMENT INFORMATION **20**
2.1. Red Hat OSP v9 Minimum Deployment Requirements 20
2.1.2. Networking Requirements *21*
2.1.3. Planning the overcloud deployment *21*

TESTING AND VALIDATION **25**
3.1. Use Case: Install and Validate F5 LBaaS *25*
3.2. Overview of the deployment and Testing *26*
3.3. Validation of the deployment Scenario *26*
3.4. Use Case Test Environment Details 27

3.4.1. Product Versions Under Test 27
3.4.2. Prerequisites for testing 27
3.4.3. Acquiring Required Products 28
3.5. PRODUCT PREPARATION FOR TEST ENVIRONMENT 29
3.5.1. Test Environment Networking Setup 29
3.6. Undercloud Installation Overview 33
3.7. Overcloud Installation Overview 35
3.8. F5 OpenStack Agent Configuration 47
4.1. Install the Testing Client 49
5.1. Items to be Tested / Not Tested 49
5.2. Multi-Tenant Community LBaaS Testing 51
5.3. Single Tenant TMOS Virtual Edition Testing 52
Items Under Test / Not Tested 54

Introduction

Today's software-defined economy requires businesses to move faster than their competitors. Speed and agility are critical to keeping up with competitive demands for new applications, as well as to maintaining existing infrastructure. IT organizations must respond aggressively to meet business needs and the private cloud can be a primary tool to achieve this objective. Not surprisingly, then, organizations are accelerating their journey to the cloud.

OpenStack for Cloud

Software developers are turning to the OpenStack platform for cloud computing. Its open APIs, flexible architecture, and large commercial ecosystem help enterprises compete in a completely new paradigm of software development. OpenStack is rapidly becoming the dominant cloud platform for delivering Infrastructure as a Service (IaaS). As OpenStack clouds increasingly host mission-critical production applications, advanced application delivery services for layers 4–7 are becoming essential. Enterprise customers deploying new applications with these services expect them to be available when they transition to a cloud-based architecture.

Planning an OpenStack Architecture

F5 is the leading supplier of advanced application delivery services across data center, public, and private clouds, including those powered by OpenStack. F5 partnered with Red Hat to help customers accelerate OpenStack deployments. OpenStack and F5 application delivery services and platforms combine to bring production-grade services to OpenStack-hosted applications. F5 application delivery services can be accessed in two ways within OpenStack: through Neutron Load Balancing as a Service v2.0 (LBaaSv2) and HEAT orchestration. With the combination of OpenStack, F5, and Red Hat, enterprises can transition from traditional data centers to private clouds faster and more efficiently.

Seamless Migration to the Cloud

The F5 private cloud solution package for OpenStack offers agile application services. These services are delivered by NetOps in an automated and continuous integration environment without compromising corporate security or reliability standards. The solution package enables efficient collaboration between DevOps/application owners and NetOps, while reducing proliferation of shadow IT. Seamless migration of applications from development to production clouds and consistent delivery of application services facilitates infrastructure-as-code initiatives. The solution package also offers simplified private cloud rollout and operational confidence with tested and certified solutions, backed by enterprise-grade support and documentation.

F5 Private Cloud Solution Package for OpenStack

The F5 private cloud solution package for OpenStack provides joint certification and testing with Red Hat to orchestrate F5® BIG-IP® Application Delivery Controllers (ADCs) with OpenStack Networking services. The validated solutions and use cases are based on customer requirements utilizing BIG-IP ADC and OpenStack integrations. F5's OpenStack LBaaSv2 integration provides under-the-cloud L4–L7 services for OpenStack Networking tenants. F5's OpenStack orchestration (HEAT) templates provide over-the-cloud, single-tenant onboarding of BIG-IP virtual edition (VE) ADC clusters and F5 iApps® templating for application services deployment.

F5 ADCs can help ease the transition as applications migrate from traditional architectures to the private cloud. The many existing applications currently utilizing F5 ADCs, application policies, and the F5 iRules® scripting language, can maintain business logic in the migration to the cloud. For other applications, BIG-IP virtual edition ADCs offer a dynamic pivot for services as they are chained and refactored into new architectures.

The OpenStack private cloud solution package documented here -- Migrating Workloads to OpenStack Private Cloud -- is the first of seven use cases. This use case, one of the most common, uses features available in existing OpenStack integrations. The F5 solution validates this use case based on tests utilizing the OpenStack integration. These tests have been validated and certified by Red Hat and published as part of F5's open source solution. This enables our customers and their partners to easily deploy and accelerate OpenStack deployment with F5 application delivery services.

Testing the solution

OpenStack installations are highly configurable and vary greatly. New OpenStack versions are released every six months, creating the need for continuous testing and validation of our solutions. This F5 solution package for the OpenStack private cloud was tested with a series of OpenStack cloud deployments and Tempest tests suites while using Red Hat OpenStack Platform (OSP) version 9, an OpenStack distribution. Red Hat, which maintains and supports the OSP, has its own test suite for LBaaSv2 certifications. F5 maintains its LBaaSv2 certification with Red Hat as part of its partnership. Red Hat also provides input and validates use case tests for private clouds against the company's own OSP cloud deployments. The successful completion of this test with Red Hat OSP version 9 forms the basis of a documented and validated solution supported by both F5 and Red Hat. See Red Hat's Customer Portal for more information on LBaaS v2.0 official support certification.

Use Case: Migrate Existing Workloads to OpenStack Private Cloud

This first use case focuses on migrating existing workloads to an OpenStack private cloud. In support of this goal, this deployment guide provides instructions for installing and validating a

Red Hat OSPv9 private cloud, then installing F5's LBaaSv2 solution. The migration presented here is based on tested deployments of Red Hat OSPv9 with F5 LBaaSv2 services, utilizing BIGIP i5800 ADC devices and a BIG-IP VE instance deployed within an OpenStack tenant. Validation and certification were performed at the F5 Labs in San Jose, California, in partnership with Red Hat. Additional information regarding the Red Hat OSP can be found in the Red Hat OpenStack Platform documentation; F5 has extensive documentation for its OpenStack solutions, in addition to the open source code on GitHub. Details on the F5 iSeries are provided on www.f5.com.

The F5 private cloud solution package for OpenStack comprises an edge deployment architecture, represented by Figure 1, below. It uses OpenStack networking provider networks with F5 agents deployed in global routed mode, as well as F5 agents deployed in L2 adjacent mode for micro-segmentation and tenant networking. The BIG-IP hardware devices in the diagram below are cloud-ready i5800 ADCs. The BIG-IP VE tenants are software ADCs, which use the F5 BIG-IQ® Centralized Management® license manager to manually license the fixed license pools and provisioning.

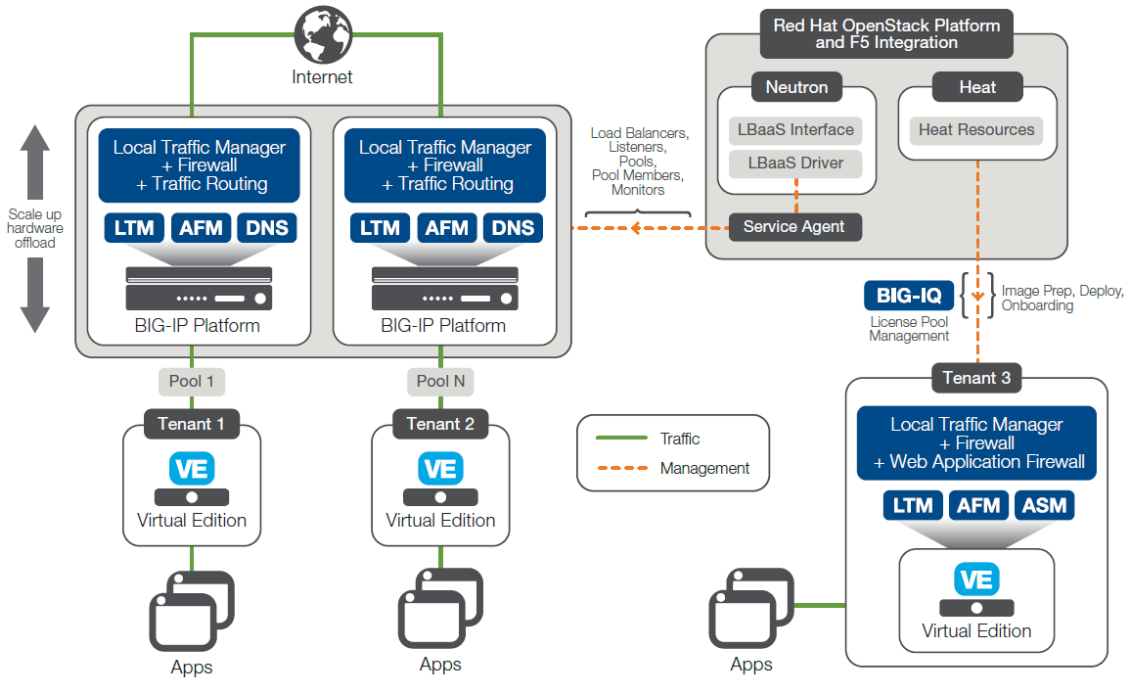

Figure 1: F5 OpenStack deployment test architecture

Tables 1 and 2 show the F5 components and build of material included in the private cloud solution package for OpenStack.

Table 1: Private Cloud Solution Packages for OpenStack: iSeries + VE + SW Solution-Engineered, Tested, and Certified

Solution Package	Medium Size
F5 iSeries	i5800 x 2
iSeries SW Modules	LTM, DNS, AFM
Virtual Edition – 200M	8
Virtual Edition – 25M	8
Virtual Edition Software Modules	LTM, ASM, AFM, Crypto
Orchestration	Heat
3rd Party Solution Certification	Red Hat OpenStack Platform (OSP) version 9
Services	40 Hour Engagement
Support	Premium Support
Customer Documentation (Customized for 5 Use Cases)	• Solution Architecture • Deployments Guide

Table 2: Private Cloud Solution Package for OpenStack: i5800M Offering and Build of Material

Components of Offering	Quantity	Detail
I5800 Better	i5800 x 2	Provides desired network packaging of LTM, DNS and AFM
200M "App Services" VE	8 (2 pools of 4 VEs)	Provides desired tenant packaging of LTM + ASM +AFM + Crypto, this packaging is only available within the Private Cloud Offering
25M "App Services" VE	8	Provides desired tenant packaging of LTM + ASM +AFM + Crypto, this packaging is only available within the Private Cloud Offering
BIG-IQ VE "S"	2	Included free as part of offering - BIG-IQ VE License Manager is needed for the VE licensing. The full Centralized Management is not needed.
iWorkflow VE "Max" – **Not needed for OpenStack**	3	Included free as part of offering
Premium Support	For all of the above	
Consulting	40 hours	1 week of consulting/scoping

About Red Hat OpenStack Platform

Red Hat OpenStack Platform (OSP) provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It offers a highly

scalable, fault-tolerant platform for the development of cloud-enabled workloads. Red Hat OpenStack Platform is packaged so that available physical hardware can be turned into a private, public, or hybrid cloud platform that includes:

- Fully distributed object storage
- Persistent block-level storage
- Virtual machine provisioning engine and image storage
- Authentication and authorization mechanisms
- Integrated networking
- Web browser-based interface accessible to users and administrators

For reference information about the components mentioned in this guide, see Deployment Information. For the complete Red Hat OpenStack Platform documentation suite, see the Red Hat OpenStack Platform Documentation.

Components

The Private Cloud Solution Package for OpenStack features multiple components from F5 and Red Hat. This chapter provides an overview of each component. The Red Hat OpenStack Platform IaaS cloud is implemented as a collection of interacting services that control compute, storage, and networking resources. The cloud can be managed with a web-based dashboard or command-line clients, which allow administrators to control, provision, and automate OpenStack resources. OpenStack also has an extensive API, which is also available to all cloud users.

Figure 2 provides a high-level overview of the OpenStack core services and their relationship with each other. Only those components pertinent to the presented use case are covered in detail in this deployment guide.

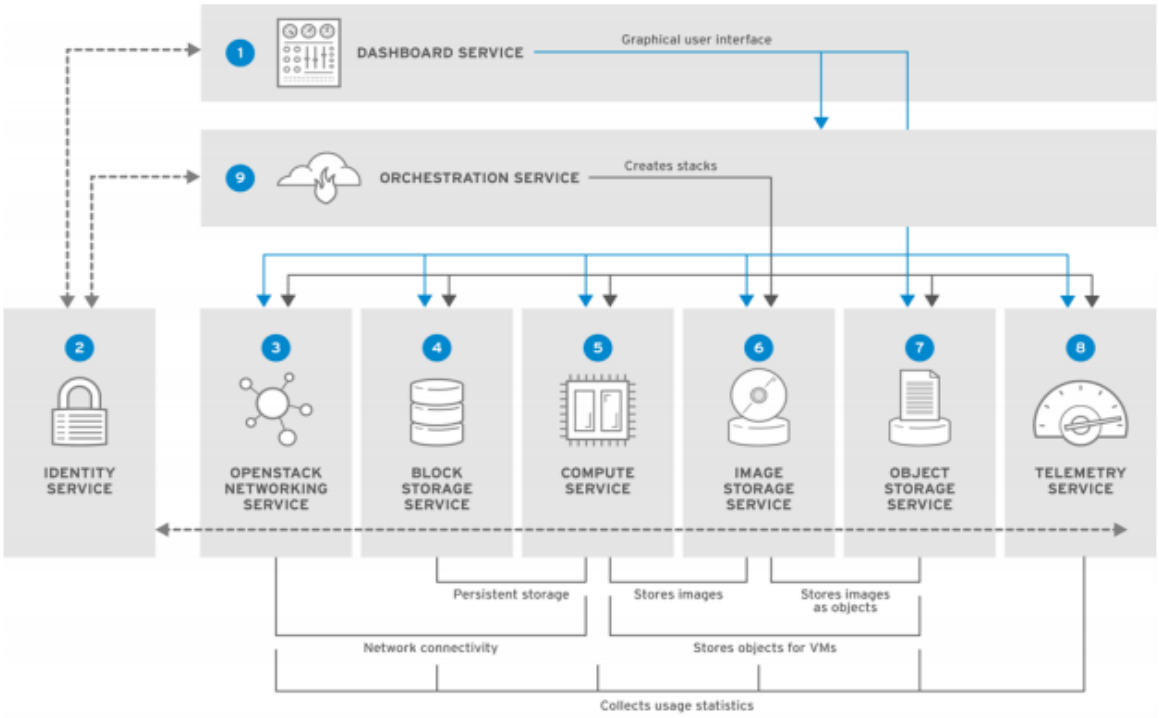

Figure 2: OpenStack components

Table 3 describes each Red Hat OpenStack Platform component shown in the diagram and provides links to the section of this document pertaining to each component.

Table 3: OpenStack components and functions

Service	Code	Description	Location
Dashboard	horizon	Web browser-based dashboard that you use to manage OpenStack services	Section 1.5.1. OpenStack Dashboard (horizon)
Identity	keystone	Centralized service for authentication and authorization of OpenStack services and for managing users, projects, and roles.	Section 1.4. Identity Management
OpenStack Networking	neutron	Provides connectivity between the interfaces of OpenStack services.	Section 1.1.1. OpenStack Networking (neutron)
Block Storage	cinder	Manages persistent block storage	Using with NFS mount

		volumes for virtual machines	
Compute	nova	Manages and provisions virtual machines running on hypervisor nodes	Section 1.3.1. OpenStack Compute (nova)
Image	glance	Registry service that you use to store resources such as virtual machine images and volume snapshots.	Section 1.3.3. OpenStack Image (glance)
Object Storage	swift	Allows users to store and retrieve files and arbitrary data.	Not used in this deployment Guide
Telemetry	ceilometer	Provides measurements of cloud resources.	Not used in this deployment Guide
Orchestration	heat	Template-based orchestration engine that supports automatic creation of resource stacks.	Section 1.3.4. OpenStack Orchestration (heat)

1.1. Networking

1.1.1. OpenStack Networking (neutron)

OpenStack Networking creates and manages a virtual networking infrastructure in the OpenStack cloud. Infrastructure elements include networks, subnets, and routers.

OpenStack Networking provides cloud administrators with the flexibility to decide which individual services to run on which physical systems. All service daemons can be run on a single physical host for evaluation purposes. Alternatively, each service can have a unique physical host, or be replicated across multiple hosts to provide redundancy. Because OpenStack Networking is software-defined, it can react in real-time to changing network needs, such as creation and assignment of new IP addresses. OpenStack Networking advantages include:

- Users can create networks, control traffic, and connect servers and devices to one or more networks.
- Flexible networking models can adapt to the network volume and tenancy.
- IP addresses can be dedicated or floating, where floating IPs can be used for dynamic traffic rerouting.

- If using VLAN networking, you can use a maximum of 4094 VLANs (4094 networks), where 4094 = 2^12 (minus 2 unusable) network addresses, which is imposed by the 12-bit header limitation.
- If using VXLAN tunnel-based networks, the VNI (Virtual Network Identifier) can use a 24-bit header, which will essentially allow around 16 million unique addresses/networks.

Table 4: OpenStack Networking components

Component	Description
Network agent	Service that runs on each OpenStack node to perform local networking configuration for the node virtual machines and for networking services such as Open vSwitch
neutron-dhcp-agent	Agent that provides DHCP services to tenant networks.
neutron-ml2	Plug-in that manages network drivers and provides routing and switching services for networking services such as Open vSwitch
neutron-server	Python daemon that manages user requests and exposes the Networking API. The default server configuration uses a plug-in with a specific set of networking mechanisms to implement the Networking API. Certain plug-ins, such as the openvswitch and linuxbridge plug-ins, use native Linux networking mechanisms, while other plug-ins interface with external devices or SDN controllers.
neutron	Command-line client to access the API

The placement of OpenStack Networking services and agents depends on the network requirements. Figure 3 shows an example common deployment model without a controller. This model utilizes a dedicated OpenStack Networking node and tenant networks.

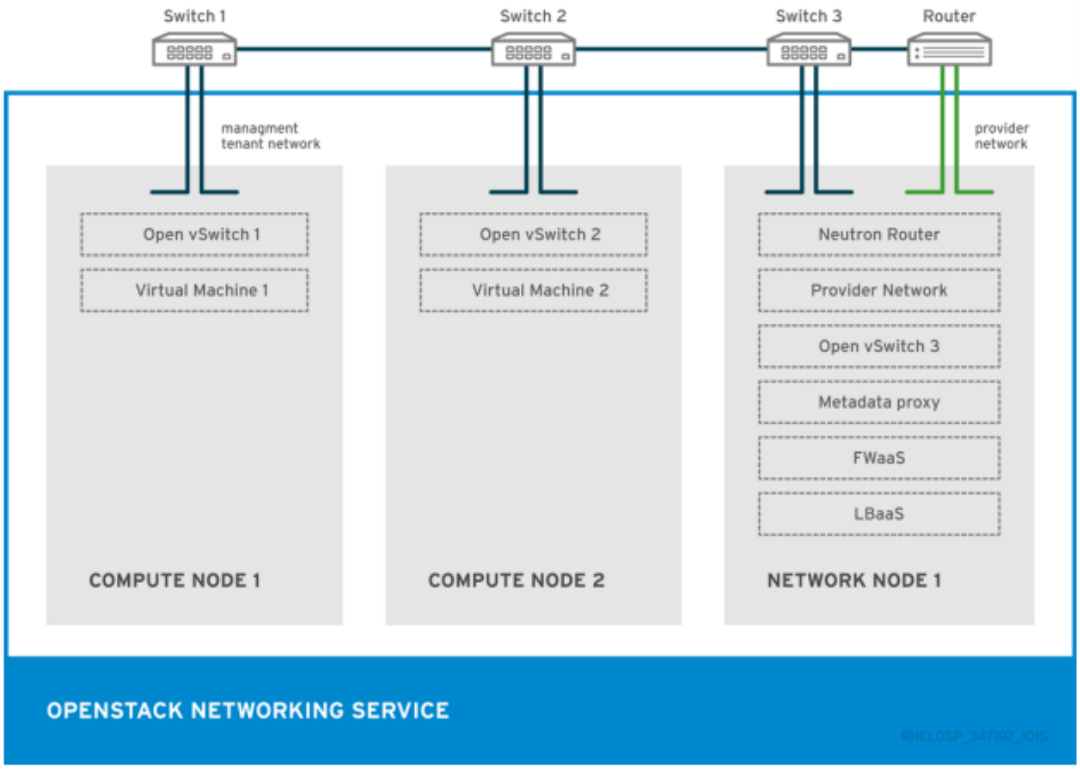

Figure 3: OpenStack Networking Components

The example shows the following Networking service configuration:

Two Compute nodes run the Open vSwitch (ovs-agent) and one OpenStack Networking node performs the following network functions:

- L3 routing
- DHCP
- NAT including services such as FWaaS and LBaaS

The compute nodes traditionally have two physical network cards each. One card handles tenant traffic and the other card manages connectivity. The OpenStack Networking node has a third network card dedicated to provider traffic. In this deployment guide, we only use a single network interface card and we do not use a dedicated network node.

1.2. Storage

Section 1.2.1. "OpenStack Block Storage (cinder)"

OpenStack Block Storage provides persistent block storage management for virtual hard drives. Block Storage allows the user to create and delete block devices and to manage attachment of block devices to servers. The actual attachment and detachment of devices is

handled through integration with the Compute service. You can use regions and zones to handle distributed block storage hosts. You can use Block Storage in performance-sensitive scenarios, such as database storage or expandable file systems. You can also use it as a server with access to raw block-level storage. Additionally, you can take volume snapshots to restore data or to create new block storage volumes. Snapshots are dependent on driver support.

OpenStack Block Storage advantages include:

- Creating, listing and deleting volumes and snapshots.
- Attaching and detaching volumes to running virtual machines

Note: In this deployment guide, we use OpenStack Block Storage Cinder with NFS (not covered in depth in this deployment guide).

Section 1.2.2. "OpenStack Object Storage (swift)"

Object Storage provides an HTTP-accessible storage system for large amounts of data, including static entities such as videos, images, email messages, files, or VM images. Objects are stored as binaries on the underlying file system along with metadata stored in the extended attributes of each file. The Object Storage distributed architecture supports horizontal scaling as well as failover redundancy with software-based data replication. Because the service supports asynchronous and eventual consistency replication, you can use it in a multiple data-center deployment.

OpenStack Object Storage advantages include:

- Storage replicas maintain the state of objects in case of outage. A minimum of three replicas is recommended.
- Storage zones host replicas. Zones ensure that each replica of a given object can be stored separately. A zone might represent an individual disk drive, an array, a server, a server rack, or even an entire data center.
- Storage regions can group zones by location. Regions can include servers or server farms that are usually located in the same geographical area. Regions have a separate API endpoint for each Object Storage service installation, which allows for a discrete separation of services.

Object Storage uses ring .gz files, which serve as database and configuration files. These files contain details of all the storage devices and mappings of stored entities to the physical location of each file. Therefore, you can use ring files to determine the location of specific data. Each object, account, and container server has a unique ring file.

Note: We do not use OpenStack Object Storage (swift) in this deployment guide.

1.3 Virtual Machines, Images and Templates

1.3.1. OpenStack Compute (nova)

OpenStack Compute serves as the core of the OpenStack cloud by providing virtual machines on demand. Compute schedules virtual machines to run on a set of nodes by defining drivers that interact with underlying virtualization mechanisms and by exposing the functionality to the other OpenStack components. Compute supports the libvirt driver -- libvirtd -- that uses KVM as the hypervisor. The hypervisor creates virtual machines and enables live migration from node to node. To provision bare metal machines, you can also review OpenStack Bare Metal Provisioning (ironic). Compute interacts with the Identity service to authenticate instance and database access; with the Image service to access images and launch instances; and with the Dashboard service to provide a user and administrator interface.

1.3.2. OpenStack Bare Metal Provisioning (ironic)

OpenStack Bare Metal Provisioning enables the user to provision physical, or bare metal, machines for a variety of hardware vendors with hardware-specific drivers. Bare Metal Provisioning integrates with the Compute service to provision the bare metal machines in the same way as virtual machines and provides a solution for the bare-metal-to-trusted-tenant use case. OpenStack Bare Metal Provisioning advantages include:

- Deploy Hadoop clusters on bare metal machines.
- Deploy hyperscale and high-performance computing (HPC) clusters.
- Use database hosting for applications that are sensitive to virtual machines.

Bare Metal Provisioning uses the Compute service for scheduling and quota management and the Identity service for authentication. Instance images must be configured to support Bare Metal Provisioning instead of KVM.

1.3.3. OpenStack Image (glance)

OpenStack Image acts as a registry for virtual disk images. Users can add new images or take a snapshot of an existing server for immediate storage. You can use the snapshots for backup or as templates for new servers. Registered images can be stored in the Object Storage service or in other locations, such as simple file systems or external Web servers.

1.3.4. OpenStack Orchestration (heat)

OpenStack Orchestration provides templates to create and manage cloud resources such as storage, networking, instances, or applications. Templates are used to create stacks, which are collections of resources. For example, you can create templates for instances, floating IPs, volumes, security groups, or users. Orchestration offers access to all OpenStack core services with a single modular template, as well as capabilities such as auto-scaling and basic high availability. OpenStack Orchestration advantages include:

- A single template provides access to all underlying service APIs.
- Templates are modular and resource-oriented.

- Templates can be recursively defined and reusable (such as nested stacks). The cloud infrastructure can then be defined and reused in a modular way.
- Resource implementation is pluggable, allowing for custom resources.
- Resources can be auto-scaled and, therefore, added or removed from the cluster based on usage.
- Basic high availability functionality is available.

Figure 4 shows the main interfaces that the Orchestration service uses to create a new stack of two new instances and a local network. This Private Cloud Solution Package uses heat for orchestration of the BIP-IP configuration. This is covered in greater detail in the "Heat Templates" section of this document.

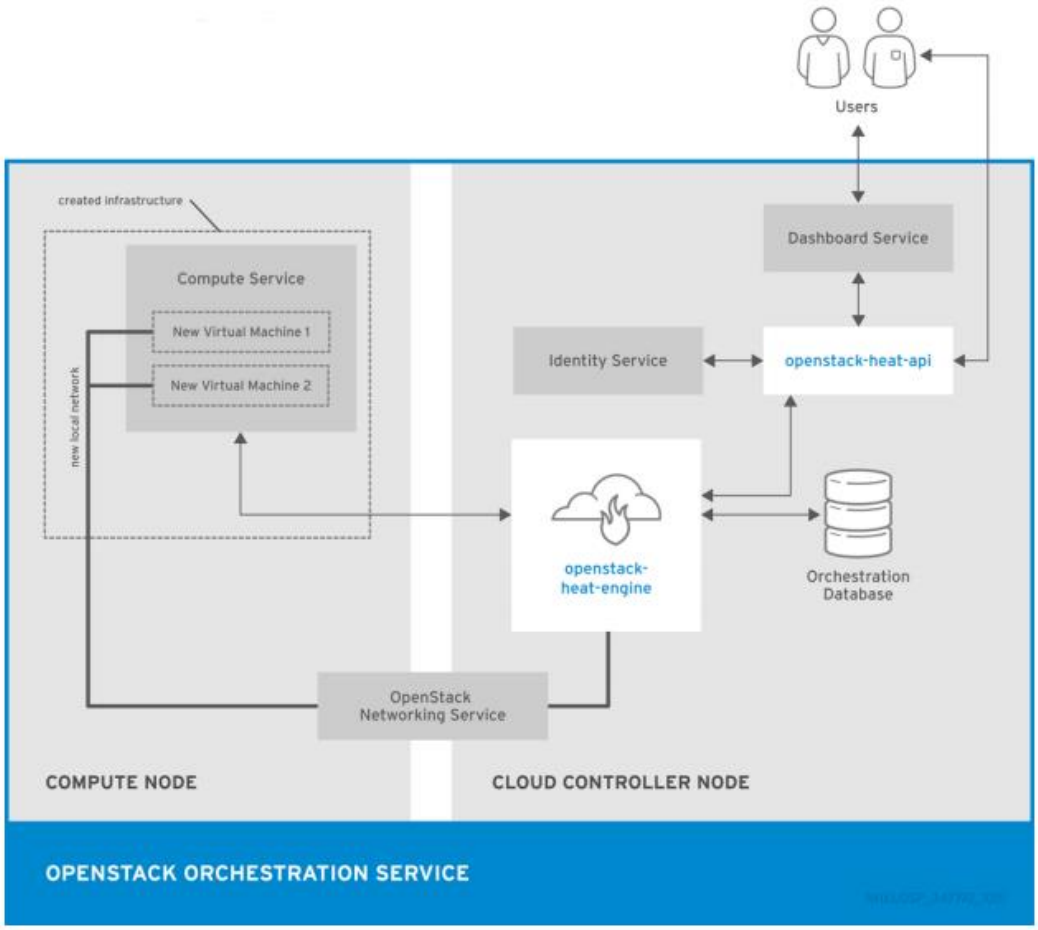

Figure 4: OpenStack Orchestration Service using Heat Templates

1.4. Identity Management

1.4.1. OpenStack Identity (keystone)

OpenStack Identity provides user authentication and authorization to all OpenStack components. Identity supports multiple authentication mechanisms, including user name and password credentials, token-based systems, and AWS-style log-ins. By default, the Identity service uses a MariaDB back end for token, catalog, policy, and identity information. This back end is recommended for development environments or to authenticate smaller user sets. You can also use multiple identity back ends concurrently, such as LDAP and SQL, and/or use memcache or Redis for token persistence. Identity supports Federation with SAML. Federated Identity establishes trust between Identity Providers (IdP) and the services that Identity provides to the end user. OpenStack Identity advantages include:

- User account management, including associated information such as a name and password. In addition to custom users, a user must be defined for each cataloged service. For example, the glance user must be defined for the Image service.
- Tenant, or project, management. Tenants can be the user group, project, or organization.
- Role management. Roles determine the user permissions. For example, a role might differentiate between permissions for a sales rep and permissions for a manager.
- Domain management. Domains determine the administrative boundaries of Identity service entities and support multi-tenancy, where a domain represents a grouping of users, groups, and tenants. A domain can have more than one tenant; if you use multiple concurrent Identity providers, each provider has one domain.

1.5. User Interfaces

1.5.1. OpenStack Dashboard (horizon)

OpenStack Dashboard provides a graphical user interface for users and administrators to perform operations such as creating and launching instances, managing networking, and setting access control.

The Dashboard service provides the Project, Admin, and Settings default dashboards. The modular design enables the dashboard to interface with other products such as billing, monitoring, and additional management tools. Figure 5 shows an example of the Compute panel in the Admin dashboard.

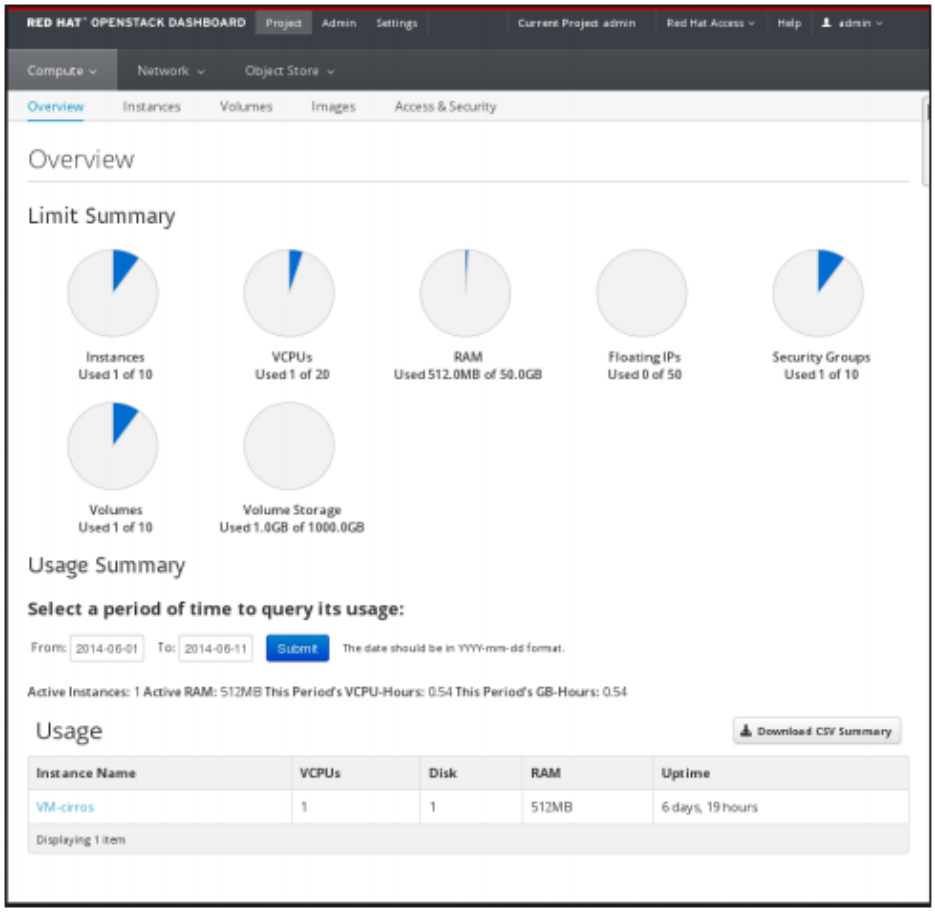

Figure 5: Horizon Dashboard

Note: In this deployment guide, we interact with this environment via CLI. The Horizon Dashboard is available to use if desired.

1.6. F5 BIG-IP Platform

F5's next-generation, cloud-ready Application Delivery Controller (ADC) platform provides DevOps-like agility with the scale, security depth, and investment protection needed for both established and emerging apps. The new F5® BIG-IP® iSeries appliances deliver quick and easy programmability, ecosystem-friendly orchestration, and record breaking, software-defined hardware performance. As a result, customers can accelerate their private clouds and secure critical data at scale while lowering TCO and future-proofing their application infrastructure.

1.6.1. The Advantages of F5 BIG-IP i5800 Hardware

The BIG-IP iSeries platform perfectly blends software and hardware innovations that balance the need for performance, scalability, and agility. The F5 TMOS® operating system provides total visibility, flexibility, and control across all application delivery services. With TMOS,

organizations can intelligently adapt to the diverse and evolving requirements of applications and networks. Other unique or patented hardware and software innovations enable the BIG-IP iSeries platform to offer unmatched capabilities:

- F5 TurboFlex™ optimization technology: Field-programmable gate arrays (FPGAs), tightly integrated with CPUs, memory, TMOS, and software, provide specific packet-flow optimizations, L4 offload, support for private cloud tunneling protocols, and denial-of-service (DoS) protection. These hardware optimizations not only improve performance, but free CPU capacity for other app delivery and security tasks as well. Only BIG-IP iSeries appliances feature TurboFlex performance profiles — user-selectable, pre-packaged optimizations that provide different performance characteristics depending on the business need.

    - L4 offload enables unsurpassed throughput and reduced loads on software.
    - Unique per-virtual-IP/application SYN flood protection ensures that if one application is under attack, others are not affected. Only F5 ADCs implement hardware-based SYN cookies in L4 and full proxy L7 mode.
    - More than 100 types of DoS attacks can be detected and mitigated in hardware, hugely increasing the attack size that can be absorbed compared to software-only implementations.
    - Network virtualization and overlay protocol processing (such as VXLAN and NVGRE tunneling) increases traffic processing capacity.
    - UDP traffic processing increases throughput and reduces both latency and jitter, improving VoIP or streaming media performance.
    - Best-in-market SSL performance accelerates SSL/TLS adoption by offloading costly SSL processing and speed key exchange and bulk encryption. BIG-IP iSeries solutions include hardware acceleration of ECC ciphers, enabling forward secrecy. In addition, the ability to achieve an SSL Labs A+ rating with one click reduces SSL configuration complexity and errors.
    - BIG-IP platforms offer maximum hardware compression, enabling cost-effective offloading of traffic compression processing to improve page load times and reduce bandwidth utilization.
    - Enterprise class SSD (solid state drive) technology on select BIG-IP platforms improves performance and reliability, saves power, and reduces heat generation and noise.
    - Efficiency features include 80 Plus Platinum certified power supplies as well as front-panel touchscreen LCD management, remote boot and multi-boot support, and USB support.

Figure 6: BIG-IP i5800 ADC appliance

1.6.2. BIG-IP Virtual Editions

F5® BIG-IP® virtual editions (VEs) are virtual application delivery controllers (vADCs) that can be deployed on all leading hypervisors and cloud platforms running on commodity servers. BIG-IP VEs deliver all the same market-leading application delivery services -- including advanced traffic management, acceleration, DNS, firewall, and access management -- as F5 purpose-built hardware. VE software images are downloadable and portable between on-premises virtualized data centers, public, and hybrid cloud environments. With BIG-IP virtual editions and F5 BIG-IQ® Centralized Management solutions, you can rapidly provision consistent application services across the data center and into the cloud.

1.6.3. BIG-IP TMOS Specifications

- The BIG-IP i5800 devices are installed with TMOS 12.1.1 licensed with LTM, DNS, AFM Services. The initial configuration should be implemented to match the deployment architecture per the F5 LBaaSv2 Installation Guide for Active/Standby HA pair configuration. Instructions for configuring the iSeries are provided in section 3.5.
- The TMOS Virtual Edition is a version 12.1.1, licensed with LTM, ASM, AFM, & Crypto Services, installed on KVM as a qcow2 image.
  Note: The image must be prepared for use in OpenStack using the supported Heat template. Instructions for using the image patch and upload template are provided in the F5 Heat Template documentation.
- Common certificate, keys, and profiles for LTM deployment are installed on the BIG-IP devices.

1.6.4. Centralized Management and Licensing with BIG-IQ

BIG-IQ Centralized Management is an intelligent framework for centrally managing F5 application delivery and security solutions. It provides a single pane of glass to manage and deploy all F5 devices, including central management for key BIG-IP modules including BIG-IP® Local Traffic Manager™ (LTM), BIG-IP® Application Security Manager™ (ASM), BIG-IP® Advanced Firewall Manager™ (AFM), BIG-IP® Access Policy Manager® (APM), as well as F5 WebSafe™. Use BIG-IQ Centralized Management for device tracking; image and configuration backup; centralized reporting and alerting; BIG-IP VE license management; and to ensure consistent security and traffic management policies across your infrastructure.

BIG-IQ's VE license management enables you to automate large-scale virtual ADC deployments in private clouds through F5 volume subscription licensing model or permanent license pools. With BIG-IQ License Manager, you can spin up and provision individual VEs, or groups of VEs, from a single license pool on demand. When resource requirements decrease, spin down the VE and return it to the license pool for future use. VE license pools are available in increments of 4 or 25.

1.6.5. F5 Virtual Edition Software Modules

- BIG-IP® Local Traffic Manager™ (LTM) VE is a virtual Application Delivery Controller. It enables you to deliver network services in a reliable, secure, and optimized way. This VNF provides Layer 4–7 load-balancing and Layer 7 traffic management—allowing you to optimize and offload other network resources, including value-added services (VAS) platforms and other VNFs in your network.
- BIG-IP® DNS VE is a virtual DNS. It secures your DNS infrastructure through high-performance DNS services, scales DNS responses to survive volume increases and distributed denial-of-service (DDoS) attacks, and ensures high availability of your global applications and services. This VNF also provides DNS scalability and delivery offload to your LDNS infrastructure. By delivering faster response times for content being accessed by fixed and mobile devices, vDNS offers higher subscriber QoE.
- F5® BIG-IP® Advanced Firewall Manager™ (AFM) is a high-performance, stateful, full-proxy network firewall designed to guard data centers against incoming threats that enter the network on the most widely deployed protocols—including HTTP/S, SMTP, DNS, and FTP. By aligning firewall policies with the applications they protect, BIG-IP AFM streamlines application deployment, security, and monitoring. With its scalability, security, and simplicity, BIG-IP AFM forms the core of the F5 application delivery firewall solution.
- BIG-IP® Application Security Manager™ (ASM) VE is a virtual web application firewall that secures web applications in traditional, virtual, and private cloud environments. It provides unmatched protection that helps secure applications against layer 7 threats including DDoS, SQL injection, OWASP Top Ten, brute force, and zero-day web application attacks. The vWAF also mitigates DOS-heavy URL attacks, prevents execution of fraudulent transactions, and stops in-browser session hijacking. With this VNF, you can achieve industry security standards compliance with key regulatory mandates.

Note: In this deployment guide, we use BIG-IQ LM to license the VE tenants manually. BIG-IQ Centralized Management and Licensing will be featured in later use cases and deployment guides.

Deployment Information

2.1. Red Hat OSP v9 Minimum Deployment Requirements

Red Hat OSP v9 is deployed in accordance with the following minimum requirements:

- 1 host machine for the Red Hat OpenStack Platform director
- 1 host machine for a Red Hat OpenStack Platform Compute node
- 1 host machine for a Red Hat OpenStack Platform Controller node

Recommended best practices for installing Red Hat OpenStack Platform is the following requirements:

- 1 host machine for the Red Hat OpenStack Platform director

- 3 host machines for Red Hat OpenStack Platform Compute nodes
- 3 host machines for Red Hat OpenStack Platform Controller nodes in a cluster
- 3 host machines for Red Hat Ceph Storage nodes in a cluster

It is recommended to use bare metal systems for all nodes. At minimum, the Compute nodes require bare metal systems. All of the Overcloud bare metal systems require an Intelligent Platform Management Interface (IPMI) because the director controls the power management. The over-the-cloud deployment configuration supports all three test architectures.

2.1.2. Networking Requirements

It is important to plan your environment's networking topology and subnets so that you can properly map roles and services to correctly communicate with each other. Red Hat OpenStack Platform uses the neutron networking service, which operates autonomously and manages software-based networks, static and floating IP addresses, and DHCP. The director deploys this service on each Controller node in an Overcloud environment.

The Undercloud host requires at least two networks:

- Provisioning network - Provides DHCP and PXE boot functions to help discover bare metal systems for use in the Overcloud. Typically, this network must use a native VLAN on a trunked interface so that the director serves PXE boot and DHCP requests. This is also the network you use for Platform Management on all Overcloud nodes.
- External Network - A separate network for remote connectivity to all nodes. The interface connecting to this network requires a routable IP address, either defined statically, or dynamically through an external DHCP service.

2.1.3. Planning the overcloud deployment

Table 5: Planning the overcloud deployment

Controller	Provides key services for controlling your environment. This includes the dashboard (horizon), authentication (keystone), image storage (glance), networking (neutron), orchestration (heat), and high availability services (if using more than one Controller node). A basic Red Hat OpenStack Platform environment requires at least one Controller node.
Compute	A physical server that acts as a hypervisor, and provides the processing capabilities required for running virtual machines in the environment. A basic Red Hat OpenStack Platform environment requires at least one Compute node.
Ceph-Storage	A host that provides Red Hat Ceph Storage. Additional Ceph Storage hosts scale into a cluster. This deployment role is optional.

Cinder-Storage	A host that provides external block storage for OpenStack's cinder service. This deployment role is optional.
Swift-Storage	A host that provides external object storage for OpenStack's Swift service. This deployment role is optional.

Red Hat OpenStack Platform maps the different services onto separate network traffic types, which are assigned to the various subnets in your environments. These network traffic types include:

Table 6: Network Type Assignments

Network Type	Description	Used By
IPMI	Network used for power management of nodes. This network is predefined before the installation of the Undercloud.	All nodes
Provisioning	The director uses this network traffic type to deploy new nodes over PXE boot and orchestrate the installation of OpenStack Platform on the Overcloud bare metal servers. This network is predefined before the installation of the Undercloud.	All nodes
Internal API	The Internal API network is used for communication between the OpenStack services using API communication, RPC messages, and database communication.	Controller, Compute, Cinder Storage, Swift Storage
Tenant	Neutron provides each tenant with their own networks using either VLAN segregation (where each tenant network is a network VLAN), or tunneling (through VXLAN or GRE). Network traffic is isolated within each tenant network. Each tenant network has an IP subnet associated with it, and network namespaces means that multiple tenant networks can use the same address range without causing conflicts.	Controller, Compute
Storage	Block Storage, NFS, iSCSI, and others. Ideally, this would be isolated to an entirely separate switch fabric for performance reasons.	All nodes
Storage Management	OpenStack Object Storage (swift) uses this network to synchronize data objects between participating replica nodes. The proxy service acts as the intermediary interface between user requests and the underlying storage layer. The proxy receives incoming requests and locates the necessary replica to retrieve the requested data. Services that	Controller, Ceph Storage, Cinder Storage, Swift Storage

	use a Ceph backend connect over the Storage Management network, since they do not interact with Ceph directly but rather use the frontend service. Note that the RBD driver is an exception, as this traffic connects directly to Ceph.	
External	Hosts the OpenStack Dashboard (horizon) for graphical system management, the public APIs for OpenStack services, and performs SNAT for incoming traffic destined for instances. If the external network uses private IP addresses (as per RFC-1918), then further NAT must be performed for traffic originating from the internet.	Controller
Floating IP	Allows incoming traffic to reach instances using 1-to-1 IP address mapping between the floating IP address, and the IP address actually assigned to the instance in the tenant network. If hosting the Floating IPs on a VLAN separate from External, you can trunk the Floating IP VLAN to the Controller nodes and add the VLAN through Neutron after Overcloud creation. This provides a means to create multiple Floating IP networks attached to multiple bridges. The VLANs are trunked but are not configured as interfaces. Instead, neutron creates an OVS port with the VLAN segmentation ID on the chosen bridge for each Floating IP network.	Controller
Management	Provides access for system administration functions such as SSH access, DNS traffic, and NTP traffic. This network also acts as a gateway for non-Controller nodes	All nodes

The diagram below provides an example of a network topology where the networks are isolated on separate VLANs. Each Overcloud node uses two interfaces (nic2 and nic3) in a bond to deliver these networks over their respective VLANs.

Note: In this deployment guide, we use a single interface.

Each Overcloud node communicates with the Undercloud over the Provisioning network through a native VLAN using nic1. The following table provides examples of network traffic mappings different network layouts. In this deployment guide, we used a single NIC set of templates. We chose not to use the default diagrams shown below. The use of single- or multi-NIC depends on your requirements and needs. Either method is good.

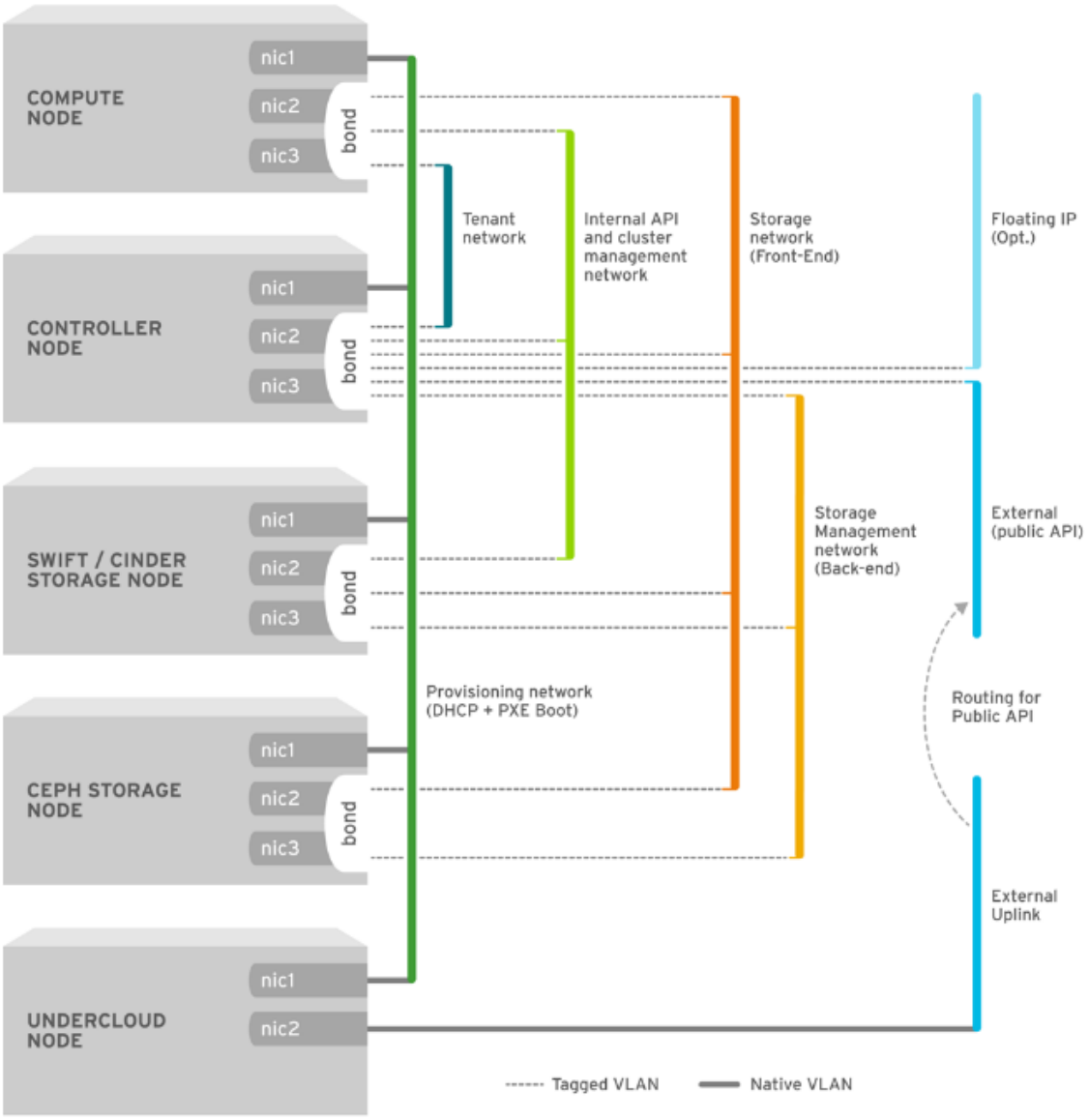

Figure 7: Network Traffic Mappings

Note: Since this is a Red Hat deployment, and not a centos box, we had to modify the instructions found on read the docs. We have asked the read the docs to reflect the proper instructions for installing onto RH OSPv9 but they are not yet updated. The read the docs want you to install PIP and to do a pip install of the f5 agents and driver. But that is not the official Red Hat way. The package installation should really be using official, Red Hat signed repositories versus other package managers, which could introduce conflicts and cause stability (and hence supportability) problems down the road in the platform itself. So, we show how to download the rpms and install them per Red hat type guidance

2.1.4. F5 OpenStack LBaaSv2 Integration

Install the F5 LBaaSv2 plugin and agent, per the Red Hat OpenStack Platform version 9 installation instructions. The agent should be configured as is appropriate to match the test architecture.

2.1.5. F5 Networks OpenStack Heat Template

The F5 OpenStack Heat templates that can be used to deploy and/or configure BIG-IP from within an OpenStack cloud can be found in the F5 Networks OpenStack Heat Template Library at http://f5-openstack-heat.readthedocs.io/en/latest/.

2.1.6. OpenStack Nova Flavors

Accommodate TMOS modules: LTM, AFM, ASM per:

The following table includes deployment references for components mentioned in this guide. Additional manuals for Red Hat OpenStack Platform can be found here: https://access.redhat.com/documentation/en-US/Red_Hat_OpenStack_Platform/

Table 7: Red Hat OpenStack Platform

Component	Description
Red Hat Enterprise Linux	Red Hat OpenStack Platform is supported on Red Hat Enterprise Linux 7.3 or later. For information on installing Red Hat Enterprise Linux, see the corresponding installation guide at: https://access.redhat.com/site/documentation/en US/Red_Hat_Enterprise_Linux/.
OpenStack	To install OpenStack components and their dependencies, use the Red Hat OpenStack Platform director. The director uses a basic OpenStack undercloud to provision and manage the OpenStack nodes in the final overcloud. **Note**: You need one additional host machine to install the undercloud. For detailed instructions, see Red Hat OpenStack Platform Director Installation and Usage.
High Availability	For the configuration of additional high availability components for BIG-IP Clustering, documented on F5.com
LBasS	To use Load Balancing-as-a-Service, see "Configuring Load Balancing-as-a-Service" documented on F5.com

Testing and Validation

3.1. Use Case: Install and Validate F5 LBaaS

This OpenStack use case assures the proper installation and operation of Neutron LBaaSv2 using the F5 service provider plugin and the F5 OpenStack agent. This plan includes LBaaSv2

testing for both global routed and L2 adjacent supported modes of operation using the F5 OpenStack agent. This deployment guide also includes testing the OpenStack Heat orchestration which compiles TMOS Virtual Edition qcow disk images for use in OpenStack.

3.2. Overview of the deployment and Testing

The tested deployment utilizes a single Red Hat OpenStack Platform version 9 cloud installation deployed with the default ML2 core network provider and Open vSwitch mechanism drivers. ML2 VLAN and VxLAN network types are used in testing.

Community produced LBaaSv2 tempest API and scenario tests are performed to validate the proper integration of F5 LBaaSv2 services with the Red Hat OpenStack Platform installation.

Only LBaaSv2 functionality supported in Red Hat OpenStack Platform version 9 are expected to pass testing. As an example, Red Hat OpenStack Platform version 9 does not support the installation of Barbican, the OpenStack secret management service. LBaaSv2 TLS offload tests depending on interaction with Barbican for certificate management are not expected to pass.

Specific community API and scenario tests will be updated when there are known limitations in the test which would preclude their use to test multi-tenant gateways. As an example, all tempest test which utilize localhost (127.0.0.1) designations which do not apply to routed gateways should be altered to use appropriate addresses.

There are specific community tests which are expected to fail because they represent violations of per-tenant isolations purposely enforced by the F5 LBaaSv2 implementation. All expected failures are enumerated.

This deployment guide also includes a simple Heat orchestration test which registers and uploads F5 BIG-IP Virtual Edition images to Glance.

3.3. Validation of the deployment Scenario

OpenStack's private cloud deployment base continues to grow in both enterprise and service provider market segments. A tested and validated Red Hat OpenStack Platform installation with F5 backed LBaaSv2 services and F5 Virtual Edition guest machine images can shorten proof-of-concept timeframes and reduce support issues.

The use of the community reference ML2 network core plugin and network types represents the obvious choice for certification testing. Other partner-backed network virtualization solutions require unique integrations which extend beyond the standard OpenStack service interfaces and core networking attributes. Unique integrations are undesirable from a base testing perspective as they introduce complexity and the need for a supportability matrix including versions of Red Hat OpenStack Platform, F5 TMOS, and partner technology solutions.

Community-based test scenarios should form the basis of all interoperability testing. This assures compliance and a wider test coverage.

In the OpenStack community, QEMU/KVM-backed compute nodes are the overwhelming preference. KVM-backed compute nodes orchestrated through libvirt-based agents are the reference implementation. Choosing TMOS Virtual Edition disk images which work with this reference environment for test is the obvious choice.

3.4. Use Case Test Environment Details

3.4.1. Product Versions under Test

- Red Hat OpenStack Services Platform version 9.
    - As of the time of publication, the Red Hat OpenStack Service Platform version 9 installs **Red Hat Enterprise Linux 7.3** running the **3.10.0-513.6.1.el7** kernel for the host operating system.
- F5 LBaaSv2 plugin driver version 9.2.0
- F5 OpenStack agent version 9.2.0
- F5 TMOS BIGIP-12.1.2.0.0.249.iso -- loaded and tested on a BIG-IP i5800 series two-member device service group. (These devices provide the target platform to service LBaaSv2 requests.)
- F5 TMOS Virtual Edition BIGIP-12.1.2.0.0.249.ALL qcow2 image - used to compile an OpenStack image and import it into Glance.
- Docker-managed container-based systems running CentOS 7.3 - the test system for the deployment.

Note: Later use cases for this deployment guide may incorporate newer versions of the F5 LBaaSv2 driver. The test plan may be updated with new versions of the cloud platform, F5 TMOS, F5 LBaaSv2 driver or agent, or F5 TMOS Virtual Edition if required to overcome a deficiency.

3.4.2. Prerequisites for testing

The minimal environment for Red Hat OpenStack Service Platform consists of four servers:

- 1 host machine for the Red Hat OpenStack Platform director
- 2 host machines for the Red Hat OpenStack Platform Compute nodes
- 1 host machine for the Red Hat OpenStack Platform Controller node

Note: At minimum, the Red Hat OpenStack Platform Compute node(s) must be physical bare metal hosts with support for Intelligent Platform Management Interface (IPMI) v2.0 management.

For this test plan, all Red Hat OpenStack Controller and Compute nodes are physical bare

metal hosts with IPMI controlled by the Red Hat OpenStack Platform Undercloud node (also known as the Director node).

The test environment includes:

- 3 physical bare metal hosts,
- 1 Red Hat OpenStack Platform Controller node, and
- 2 Red Hat OpenStack Platform Compute nodes.
-

The minimum requirements for all physical bare metal hosts are:

- 8-core 64bit x86 processor
- 16 GB RAM
- 40 GB physical disk.

The LBaaSv2 service configures an active-standby pair of BIG-IP i5800 series appliances with the following requirements:

- A total of 18 VLAN-based network segments
- A locally hosted `BIGIP-12.1.2.0.0.249.ALL.qcow2` image

Note: If you follow the steps presented here, you will have a functioning cloud suitable for proof of concept or testing. The only true requirement for installing the F5 components, though, is a cloud that passes the network Tempest tests. This means that customers can build their own clouds in their own way; as long as the cloud passes the Tempest tests, it qualifies for installation of the F5 LBaaSv2 driver and agent.

3.4.3. Required Products

Red Hat OpenStack Platform is available via support subscription from Red Hat. A valid subscription license with the following feature attributes will be required:

- Red Hat Enterprise Linux Server
- Red Hat OpenStack
- Red Hat OpenStack Certification Test Suite

Documentation for Red Hat OpenStack Platform version 9 is available online at https://access.redhat.com/documentation/en/red-hat-openstack-platform?version=9/

The F5 TMOS installation ISO compatible with iSeries appliances can be downloaded from https://downloads.f5.com. iSeries appliances require a license that includes the LTM and SDN Services features flags.

The F5 TMOS Virtual Edition qcow disk images compatible with KVM hypervisor can also be found at https://downloads.f5.com.

Documentation for F5 TMOS 12.1.2 is available online, at https://support.f5.com/csp/tech-documents and at askf5.com.

3.5. Product Preparation for Test Environment

3.5.1. Test Environment Networking Setup

The test plan uses the networking setup shown in Table 8.

Table 8: Use Case Environment Network Mapping

Network VLAN Tag	Network IP CIDR	Description	Connected Hosts
156	1.1.1.0/24	BIG-IP HA	BIG-IP iSeries
180	10.105.180.0/24	Provisioning Network	Undercloud Node Controller Node Compute Nodes BIG-IP iSeries
181	10.105.181.0/24	External Network	Undercloud Node Controller Node BIG-IP iSeries
182	10.105.182.0/24	Internal API Network	Controller Node Compute Nodes
183	10.105.183.0/24	Tenant Underlay Network	Controller Node Compute Nodes BIG-IP iSeries
184	10.105.184.0/24	Storage Network	Controller Node Compute Nodes
186	10.105.186.0/24	Storage Management Network	Controller Node

187	10.105.187.0/24	Overcloud Provider Network	Controller Node Compute Nodes BIG-IP iSeries
188	10.105.188.0/24	Overcloud Provider Network	Controller Node Compute Nodes BIG-IP iSeries
220-229	Dynamic	Tenant VLAN Networks	Controller Node Compute Nodes BIG-IP iSeries
192	192.168.0.0/16	IPMI	Controller Node IPMI Compute Node IPMI

Note: In our environment, each of these networks can route to any of the other network segments.

Figure 9 is a basic connectivity diagram for all systems defined in this test plan.

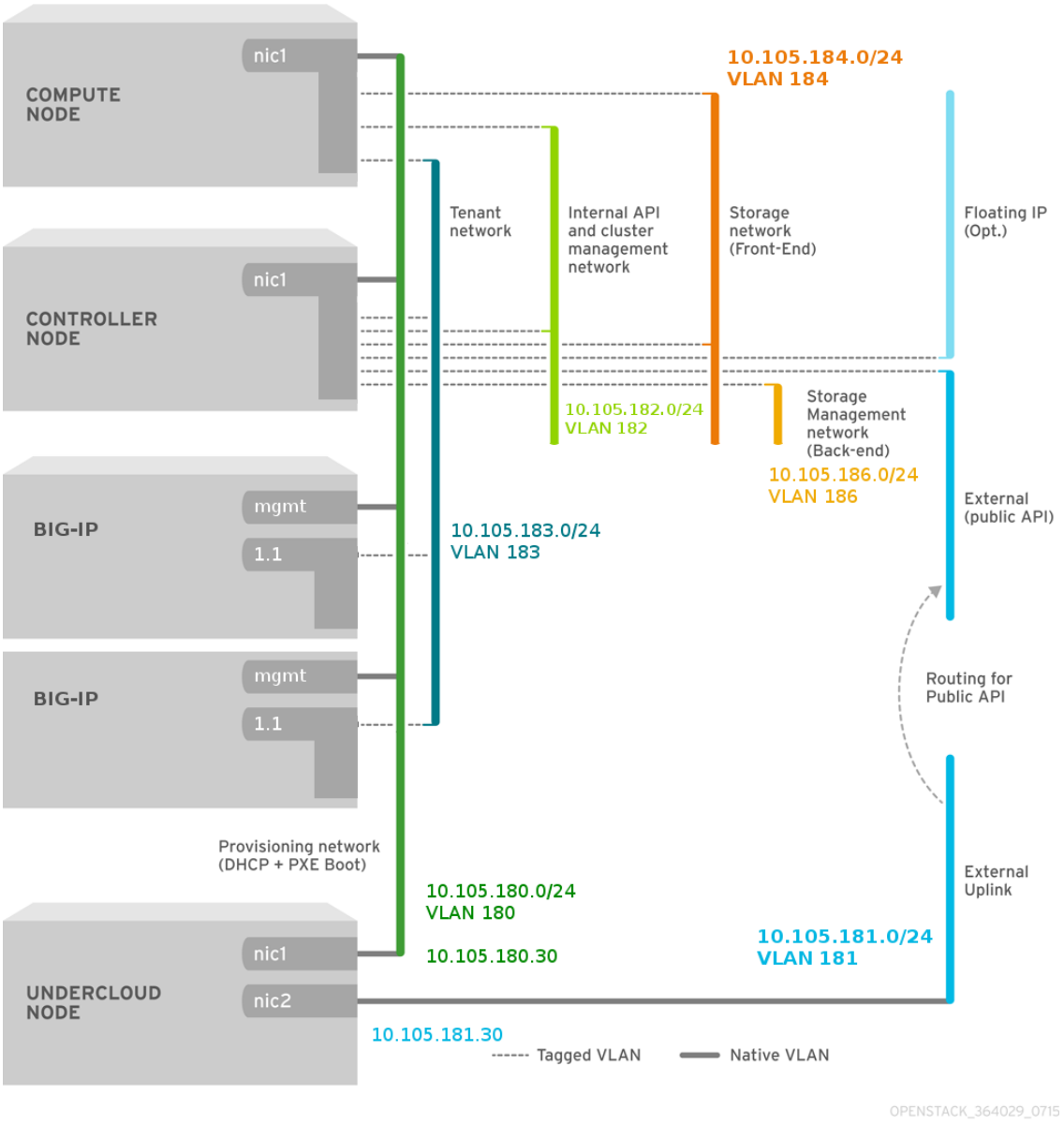

Figure 9: Use Case Environment Network Mapping

Note: This test deployment is designed to test networking service functionality. The storage deployment has therefore been simplified to use the minimal supportable configuration. This configuration is not intended to represent a production deployment. Rather, it represents the minimum deployment necessary to test the F5 Networks OpenStack LBaaSv2 driver and agent.

3.5.3. Undercloud Node

Allocate one Intel x64 base system as the undercloud node (also referred to as the OSP director node). This system **does not** have to be a bare metal server, although it is only supported on RHEL 5-7 KVM, Red Hat Virtualization 3.0-3.6,4.0, Microsoft Hyper-V, and VMWare ESX and ESXi as a virtualized undercloud.

Important: This system must have a single disk; otherwise, it will not work with the documented undercloud deployment.

The undercloud, or director, node requires two network interfaces. The first network interface must support promiscuous mode operation. If deploying the Undercloud node on a VMware ESXi hypervisor, as we have chosen to do, additional details on the provisioning network interface configuration can be found here: https://access.redhat.com/solutions/1980283

This system should be built with RHEL 7.3 and have a valid subscription.

Do not use the default disk layout for RHEL 7.3. This system must only include a `/boot`, a `/(root)`, and `swap` partition. The undercloud deployment requires that only these three partitions be present.

As an example, a SATA drive can be partitioned to show:

```
            swap 16 G
/dev/sda1 - /boot  512M
/dev/sda3 - / (root) remainder of disk
```

Configuration Steps:

1. Configure the first network interface as an untagged interface on VLAN 180.
2. Add the static IP address 10.105.180.30 to the interface.
3. Configure the second network interface as an untagged interface on VLAN 181.
4. Assign the static IP address 10.105.181.30 to the interface.
5. Provision the system default gateway as 10.105.181.1.

3.5.4. Controller Host Systems

Allocate one Intel x64-based bare metal server as the Controller Node.

Important: This system must have a single disk; otherwise, it will not work with the documented undercloud deployment.

The system uses a single network interface.

Configuration Steps:

1. Connect the network interface to the switch port.
2. Set the native VLAN to VLAN 180.
3. Allow VLANs 181-184, 186-188, & 220-229.

3.5.5. Compute Host Systems

Allocate two Intel x64-based bare metal servers as Compute Nodes.

Important: Each system must have a single disk; otherwise, they will not work with the documented undercloud deployment.

Each Compute Node uses a single network interface. Complete the configuration steps below for each Compute Node.

Configuration Steps:

1. Connect the network interface to the switch port.
2. Set the native VLAN to VLAN 180.
3. Also allow VLANs 182-184, 187-188, 220-229.

3.5.6. BIG-IP iSeries Systems

This deployment requires two BIG-IP iSeries appliances; i5800 series appliances have been tested for the purposes of this document.

Configuration Steps:

1. Connect the **management interface** to a port with the native VLAN set to VLAN 180.
2. Connect **data interface 1.1** to a port and allow VLANs 181,183, 187-188, & 220-229.

3.5.7. Test System

This deployment requires an Intel x64 system with CentOS 7.3 and Docker installed.

The test system should have a single network interface.

Configuration Steps:

- Connect the interface to a port with the native VLAN set to VLAN 181.
- Assign a static IP address, 10.105.181.220, to the interface.
- Provision the system default gateway as 10.105.181.1.

3.6. Undercloud Installation Overview

Note: The instructions provided here contain information originally published in the Red Hat OpenStack Platform Undercloud Installation and Usage guide: https://access.redhat.com/documentation/en/red-hat-openstack-platform/9/paged/director-installation-and-usage/

If installing the Undercloud on ESXi, see https://access.redhat.com/solutions/1980283

3.6.1. Undercloud Installation

This section summarizes Chapter 4 – Installing the Undercloud of the *Director Usage and Installation Guide*. The procedures in the *Director Usage and Installation Guide* for Red Hat OpenStack Platform version 9 should be carefully followed.

The undercloud installation inventories and creates RHEL 7.3 disk images for the Controller Node and the Compute Nodes. All bare metal servers' IPMI interfaces must be reachable from the undercloud node. The undercloud node uses IPMI to force reboots and issue power control commands to the bare metal servers.

IPMI access is defined in an inventory JSON file. The contents of the host inventory JSON file describing the test deployment are shown below:

```
{
        "nodes": [{
                "pm_type": "pxe_ipmitool",
                "pm_user": "ADMIN",
                "pm_password": "ADMIN",
                "pm_addr": "192.168.199.5"
        }, {
                "pm_type": "pxe_ipmitool",
                "pm_user": "ADMIN",
                "pm_password": "ADMIN",
                "pm_addr": "192.168.199.6"
        }, {
                "pm_type": "pxe_ipmitool",
                "pm_user": "ADMIN",
                "pm_password": "ADMIN",
                "pm_addr": "192.168.199.7"
        }]
}
```

Note: In our environment, the 192.168.199.0/24 subnet is routable from the 10.105.181.0/24 network.

The undercloud installation uses OpenStack Ironic services to inventory and assign roles, then uses OpenStack Heat orchestration to build out the bare metal servers in their assigned roles.

The undercloud process uses addresses 10.105.180.40-59 within the provisioning network CIDR.

- Addresses 10.105.180.40-49 are used to allocate provisioning network IP addresses for the Controller node and Compute nodes.
- Addresses 10.105.180.50-59 are used for inventory and inspection services on the provisioning network.

The content of the uncommented lines of our *undercloud.conf* file for the testing deployment is shown below:

```
[DEFAULT]
local_ip = 10.105.180.30/24
network_gateway = 10.105.180.30
local_interface = ens192
network_cidr = 10.105.180.0/24
masquerade_network = 10.105.180.0/24
dhcp_start = 10.105.180.40
dhcp_end = 10.105.180.49
inspection_iprange = 10.105.180.50,10.105.180.59
undercloud_debug = true
```

Note: It is strongly recommended that iPXE be used for the installation. With iPXE HTTP is used to fetch the boot images for the bare metal servers instead of TFTP. This should be the default anyway (if one comments out ipxe_true, it should default to ipxe).

Reboot at the end of the Undercloud installation to ensure that all services come up correctly. Issues with the openstack overcloud image upload command not working because keystone or glance were not fully up have been reported. A reboot will resolve these issues. Additional information is provided at https://bugzilla.redhat.com/show_bug.cgiid=1383509

Restart and validate the `openstack-ironic-inspector-dnsmasq` service using the following commands:

```
sudo systemctl restart openstack-ironic-inspector-dnsmasq
sudo systemctl status openstack-ironic-inspector-dnsmasq
```

Upon finishing the instructions in Chapter 4 – Installing the Undercloud of the *Director Usage and Installation Guide*, you should have the following:

- one bare metal server registered as an Ironic node with profile *control* listed in its `instance_info` attribute.
- two bare metal servers registered as Ironic nodes with profile *compute* listed in their `instance_info` attribute.

3.6.2. TMOS Undercloud Installation

Important: The F5 BIG-IP iSeries appliances do not support TMOS installation via undercloud deployment orchestration. TMOS 12.1.2 should be installed on the BIG-IP iSeries devices using TMOS software management interfaces.

Each BIG-IP iSeries appliance should be licensed with the LTM and SDN Services feature flags **before** beginning the overcloud installation.

3.7. Overcloud Installation Overview

3.7.1. Overcloud Installation

This section summarizes information from Chapter 5 – Configuring Basic Overcloud Requirements of the *Director Usage and Installation Guide*. Skip section 5.4, as we specified a single disk on the compute and controller nodes.

The overcloud single NIC deployment Heat templates were used with the following environment files, created to match the use case deployment.

Table 9: Overcloud Installation Environment File Name

Environment File Name	Description
global-defaults.yaml	Define all global template input parameters
network-environment.yaml	Define all VLANs and subnet allocations
storage-environment.yaml	Define the minimal NFS based storage deployment
service-net-map.yaml	Allow access to admin Identity services from the external network for test client access

Table 10 summarizes the overcloud environment file.

Table 10: Overcloud Installation Environment File Contents

global-defaults.yaml - parameter_defaults	
`TimeZone: 'US/Pacific'`	Environment timezone
`NtpServer: 'pool.ntp.org'`	Set NTP server to your environment's NTP source
`ControllerCount: 1`	Deploy only 1 Controller node
`ComputeCount: 2`	Deploy two Compute nodes
`ObjectStorageCount: 0`	Do not deploy any dedicated Swift nodes
`BlockStorageCount: 0`	Do not deploy any dedicated Cinder nodes
`CephStorageCount: 0`	Do not deploy an Ceph storage backing nodes
`Debug: 'true'`	Turn on debug logging for the overcloud deployment

The output of our sample */home/stack/templates/global-defaults.yaml* is below.

```
[stack@rh-director-09 ~]$ cat templates/global-defaults.yaml
parameter_defaults:
  TimeZone: 'US/Pacific'
  NtpServer: 'pool.ntp.org'
  ControllerCount: 1
  ComputeCount: 2
  Debug: 'true'
  ObjectStorageCount: 0
  BlockStorageCount: 0
  CephStorageCount: 0
```

Table11: Overcloud Network Environments Yaml file

network-environment.yaml - parameter_defaults	

`ExternalNetworkVlanID: 181`
`ExternalNetCidr: 10.105.181.0/24`
`ExternalAllocationPools: [{'start': '10.105.181.40', 'end': '10.105.181.100'}]`
`NeutronExternalNetworkBridge: "''"`
`ExternalInterfaceDefaultRoute: 10.105.181.1`
	External network definitions

`InternalApiNetworkVlanID: 182`
`InternalApiNetCidr: 10.105.182.0/24`
`InternalApiAllocationPools: [{'start': '10.105.182.40', 'end': '10.105.182.100'}]`
	Internal API access network definitions

`TenantNetworkVlanID: 183`
`TenantNetCidr: 10.105.183.0/24`
`TenantAllocationPools: [{'start': '10.105.183.40', 'end': '10.105.183.100'}]`
	Tenant network definitions

`StorageNetworkVlanID: 184`
`StorageNetCidr: 10.105.184.0/24`
`StorageAllocationPools: [{'start': '10.105.184.40', 'end': '10.105.184.100'}]`
	Storage network definitions

`StorageMgmtNetworkVlanID: 186`
`StorageMgmtNetCidr: 10.105.186.0/24`
`StorageMgmtAllocationPools: [{'start': '10.105.186.40', 'end': '10.105.186.100'}]`
	Storage management network definitions

`ControlPlaneDefaultRoute: 10.105.180.30` `EC2MetadataIp: 10.105.180.30`	Control plane default routes and metadata proxy definitions
`DnsServers: ["10.105.134.20","10.105.134.21"]`	DNS server definitions

The output of our sample */home/stack/templates/network-environment.yaml* file is shown below.

```
[stack@rh-director-09 ~]$ cat templates/network-environment.yaml
# This template configures each role to use Vlans on a single nic for
# each isolated network.
# This template assumes use of network-isolation.yaml.
#
# FIXME: if/when we add functionality to heatclient to include heat
# environment files we should think about using it here to automatically
# include network-isolation.yaml.
resource_registry:
  # OS::TripleO::BlockStorage::Net::SoftwareConfig: /home/stack/templates/nic-configs/cinder-
storage.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/templates/nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/templates/nic-configs/controller.yaml
  # OS::TripleO::ObjectStorage::Net::SoftwareConfig: /home/stack/templates/nic-configs/swift-
storage.yaml
  # OS::TripleO::CephStorage::Net::SoftwareConfig: /home/stack/templates/nic-configs/ceph-storage.yaml

parameter_defaults:
  InternalApiNetCidr: 10.105.182.0/24
  TenantNetCidr: 10.105.183.0/24
  StorageNetCidr: 10.105.184.0/24
  StorageMgmtNetCidr: 10.105.186.0/24
  # ManagementNetCidr: 10.105.180.0/24
  ExternalNetCidr: 10.105.181.0/24
  InternalApiAllocationPools: [{'start': '10.105.182.40', 'end': '10.105.182.100'}]
  TenantAllocationPools: [{'start': '10.105.183.40', 'end': '10.105.183.100'}]
  StorageAllocationPools: [{'start': '10.105.184.40', 'end': '10.105.184.100'}]
  StorageMgmtAllocationPools: [{'start': '10.105.186.40', 'end': '10.105.186.100'}]
  # ManagementAllocationPools: [{'start': '10.105.180.101', 'end': '10.105.180.150'}]
  # Leave room for floating IPs in the External allocation pool
  ExternalAllocationPools: [{'start': '10.105.181.40', 'end': '10.105.181.100'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 10.105.181.1
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 10.105.180.30
  # The IP address of the EC2 metadata server. Generally the IP of the Undercloud
  EC2MetadataIp: 10.105.180.30
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["10.105.134.20","10.105.134.21"]
  InternalApiNetworkVlanID: 182
  StorageNetworkVlanID: 184
  StorageMgmtNetworkVlanID: 186
  TenantNetworkVlanID: 183
  # ManagementNetworkVlanID: 180
  ExternalNetworkVlanID: 181
  # Set to "br-ex" if using floating IPs on native VLAN on bridge br-ex
  NeutronExternalNetworkBridge: "''"

  # If you are using VLANS as the network type rather than VXLAN or GRE,
  # then uncomment the network type vlan line and the VLANRanges line
  # the VLANRanges is in format of
  # <physical network>:<start 802.1q ID>:<end 802.1q ID>
  # NeutronNetworkType: vlan
  # NeutronNetworkVLANRanges: datacentre:220:229
  #
  #
  # If you are using VXLAN as the network type rather than VLAN or GRE
  # uncomment the following three lines and specify your start and end VNIs
  NeutronNetworkType: vxlan
  NeutronVniRanges: 5000:5800
         NeutronTunnelTypes: vxlan
```

Note: We have included the ability to change the overcloud from VXLAN-based to VLAN-based by simply uncommenting and commenting the lines above as needed for the desired deployment. In this environment, we have validated both VXLAN- and VLAN-based solutions.

Table 12: Overcloud Storage Environment Yaml Content

storage-environment.yaml - parameter_defaults	
CinderEnableIscsiBackend: false CinderEnableRbdBackend: false CinderEnableNfsBackend: true CinderNfsServers: '10.105.149.40:/software'	Cinder storage definitions
NovaEnableRbdBackend: false	Nova storage definitions
GlanceBackend: file	Glance image storage definitions
GnocchiBackend: file	Ceilometer metric storage definitions

The output of our sample */home/stack/templates/storage-environment.yaml* is shown below.

```
[stack@rh-director-09 ~]$ cat templates/storage-environment.yaml
## A Heat environment file which can be used to set up storage
## backends. Defaults to Ceph used as a backend for Cinder, Glance and
## Nova ephemeral storage.
parameter_defaults:

  #### BACKEND SELECTION ####

  ## Whether to enable iscsi backend for Cinder.
  CinderEnableIscsiBackend: false
  ## Whether to enable rbd (Ceph) backend for Cinder.
  CinderEnableRbdBackend: false
  ## Whether to enable NFS backend for Cinder.
  CinderEnableNfsBackend: true
  ## Whether to enable rbd (Ceph) backend for Nova ephemeral storage.
  NovaEnableRbdBackend: false
  ## Glance backend can be either 'rbd' (Ceph), 'swift' or 'file'.
  GlanceBackend: file
  ## Gnocchi backend can be either 'rbd' (Ceph), 'swift' or 'file'.
  GnocchiBackend: file


  #### CINDER NFS SETTINGS ####

  ## NFS mount options
  # CinderNfsMountOptions: ''
  ## NFS mount point, e.g. '192.168.122.1:/export/cinder'
  CinderNfsServers: '10.105.149.40:/software'


  #### GLANCE FILE BACKEND PACEMAKER SETTINGS (used for mounting NFS) ####

  ## Whether to make Glance 'file' backend a mount managed by Pacemaker
  # GlanceFilePcmkManage: false
  ## File system type of the mount
  # GlanceFilePcmkFstype: nfs
  ## Pacemaker mount point, e.g. '192.168.122.1:/export/glance' for NFS
  # GlanceFilePcmkDevice: ''
  ## Options for the mount managed by Pacemaker
  # GlanceFilePcmkOptions: ''

  #### CEPH SETTINGS ####

  ## Whether to deploy Ceph OSDs on the controller nodes. By default
  ## OSDs are deployed on dedicated ceph-storage nodes only.
  # ControllerEnableCephStorage: false

  ## When deploying Ceph Nodes through the oscplugin CLI, the following
  ## parameters are set automatically by the CLI. When deploying via
  ## heat stack-create or ceph on the controller nodes only,
  ## they need to be provided manually.

  ## Number of Ceph storage nodes to deploy
  # CephStorageCount: 0
  ## Ceph FSID, e.g. '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # CephClusterFSID: ''
  ## Ceph monitor key, e.g. 'AQC+Ox1VmEr3BxAALZejqeHj50Nj6wJDvs96OQ=='
  # CephMonKey: ''
  ## Ceph admin key, e.g. 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  # CephAdminKey: ''
```

The following file was copied from */usr/share/openstack-tripleo-heat-templates/overcloud-without-mergepy.yaml*. See https://access.redhat.com/solutions/2891731 for information on this change.

Create a custom `service-net-map` environment file and update the `KeystoneAdminApiNetwork` parameter.

Note: This environment file must include all services; the default parameters can be found in */usr/share/openstack-tripleo-heat-templates/overcloud-without-mergepy.yaml*

Create the file *~stack/templates/service-net-map.yaml* with the default `KeystoneAdminApiNetwork` parameter described in Table 13.

Table 13: Overcloud Service Map Yaml Content

service-net-map.yaml - parameter_defaults	
KeystoneAdminApiNetwork: external	Allow Identity service access for project and user creation from the external network.

The following is the output of our sample */home/stack/templates/service-net-map.yaml*:

```
[stack@rh-director-09 templates]$ cat service-net-map.yaml
parameter_defaults:
  ServiceNetMap:
    NeutronTenantNetwork: tenant
    CeilometerApiNetwork: internal_api
    AodhApiNetwork: internal_api
    GnocchiApiNetwork: internal_api
    MongoDbNetwork: internal_api
    CinderApiNetwork: internal_api
    CinderIscsiNetwork: storage
    GlanceApiNetwork: storage
    GlanceRegistryNetwork: internal_api
    KeystoneAdminApiNetwork: external
    KeystonePublicApiNetwork: internal_api
    NeutronApiNetwork: internal_api
    HeatApiNetwork: internal_api
    NovaApiNetwork: internal_api
    NovaMetadataNetwork: internal_api
    NovaVncProxyNetwork: internal_api
    SwiftMgmtNetwork: storage_mgmt
    SwiftProxyNetwork: storage
    SaharaApiNetwork: internal_api
    HorizonNetwork: internal_api
    MemcachedNetwork: internal_api
    RabbitMqNetwork: internal_api
    RedisNetwork: internal_api
    MysqlNetwork: internal_api
    CephClusterNetwork: storage_mgmt
    CephPublicNetwork: storage
    ControllerHostnameResolveNetwork: internal_api
    ComputeHostnameResolveNetwork: internal_api
    BlockStorageHostnameResolveNetwork: internal_api
    ObjectStorageHostnameResolveNetwork: internal_api
    CephStorageHostnameResolveNetwork: storage
```

Using the single-NIC deployment templates requires inclusion of the *network-isolation.yaml* environment file as part of the overcloud deployment command. If you do not include the *network-isolation.yaml* file, the network resources will not reference the correct Heat templates.

In our environment, we created a script to run the OpenStack overcloud deploy command, so we wouldn't have to retype all the parameters we needed each time. It is located at */home/stack/deploy_overcloud.sh* and shown below.

```
[stack@rh-director-09 ~]$ cat deploy_overcloud.sh
#!/bin/bash
openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
  -e ~/templates/network-environment.yaml \
  -e ~/templates/storage-environment.yaml \
  -e ~/templates/global-defaults.yaml \
  -e ~/templates/service-net-map.yaml \
  --compute-flavor compute \
  --control-flavor control \
  --ceph-storage-flavor ceph-storage \
```

Start a second SSH session and source the */home/stack/stackrc* file, then run the following to monitor the heat stacks that are in progress. When the watch command shows no output, the OpenStack `overcloud deploy` command has finished.

```
[stack@rh-director-09 ~]$ watch "heat stack-list --show-nested | grep -v COMPLETE"
```

3.7.2. Post-deployment steps

As directed in chapter 8 – Performing Tasks After Overcloud Creation of the *Director Usage and Installation Guide*:

- Create the `heat_stack_owner` role before running any heat orchestrations or heat tests. This role must be added to the admin user in the admin project. You can use the script below to create the role and add it to the correct user and project.

```
#!/bin/bash
source /home/stack/overcloudrc
if ! openstack role show heat_stack_owner > /dev/null 2>&1; then
   openstack role create heat_stack_owner
   openstack role add --project admin --user admin heat_stack_owner
fi
```

- Add external network

```
neutron net-create public --router:external --provider:network_type vlan --
provider:physical_network datacentre --provider:segmentation_id 181
neutron subnet-create --name public --enable_dhcp=False --allocation-
pool=start=10.105.181.101,end=10.105.181.199 --gateway=10.105.181.1 public
10.105.181.0/2
openstack router create admin_router
```

This creates the router and shows its ID.

```
openstack network list
+--------------------------------------+---------------------+--------------------------------------+
| ID                                   | Name                | Subnets                              |
+--------------------------------------+---------------------+--------------------------------------+
| c63fef57-dfce-4ae8-a3bf-cedbf7c58386 | provider_network_187 | fd7cd3c5-d1e0-45b4-be4a-f045e6d43e2f |
| 037fbd8d-865f-4953-a1b6-bde6699f4c03 | default              | c015c9e1-70aa-4ed5-a09e-17facd3ae1be |
| ce7c26ad-9a45-48d1-bfa5-513268a81681 | provider_network_188 |                                      |
| 8c0a8f81-8496-4a16-9f96-103f05c8a0c3 | public               | adf61306-a124-4444-b7a7-df0caad2f3c9 |
+--------------------------------------+---------------------+--------------------------------------+
```

- Copy the router ID and the public network ID into the command below:

```
neutron router-gateway-set 636e4c62-4949-4820-9fb3-725397d43916 8c0a8f81-8496-
4a16-9f96-103f05c8a0c3
```

- Create the following provider networks in preparation for Global Routed Mode tests:

```
neutron net-create --provider:physical_network datacentre --provider:network_type
vlan --provider:segmentation_id 187 provider_network_187
neutron net-create --provider:physical_network datacentre --provider:network_type
vlan --provider:segmentation_id 188 provider_network_188
neutron subnet-create --name provider-subnet-187 --enable_dhcp=True --allocation-
pool start=10.105.187.50,end=10.105.187.100 --gateway 10.105.187.254
provider_network_187 10.105.187.0/24
```

All overcloud templates and environment files are achieved as part of the published test results.

Chapter 8, section 8.5, illustrates running Tempest tests to validate the overcloud deployment. Since the test cloud deployment focuses solely on network services, it is recommended that you run just the network test:

```
tools/run_tests.sh '^tempest.api.network'
```

The Tempest network test validates the overcloud deployment. If your deployment passes the Tempest tests, you can install the F5 LBaaSv2 components.

Bear in mind that we've demonstrated only one of many possible ways to deploy a Red Hat OSP v9 overcloud. The key takeaway is that your deployment must pass the Tempest tests before you can install the F5 LBaaSv2 components.

3.7.3. TMOS Overcloud Installation

First, configure the BIG-IP iSeries devices to perform multi-tenant LBaaS functions for the overcloud installation.

The networking and IP addresses applied to the BIG-IP iSeries devices in our test configuration are shown in Table 14.

Table 14: BIG-IP Configuration Variables

Device	Interface	VLAN ID	VLAN Name	Address	Self IP Name
A	management			10.105.180.200/24	
A	1.1 tagged	156	/Common/HA	1.1.1.1/24	/Common/HA
A	1.1 tagged	183	/Common/Tenant183	10.105.183.101	/Common/vtep
B	management				
B	1.1 tagged	156	/Common/HA	1.1.1.2/24	/Common/HA
B	1.1 tagged	183	/Common/Tenant183	10.105.183.102	/Common/vtep

The BIG-IP iSeries appliances must be peered in a **sync-failover device service group** for proper high-availability failover.

- Run the following script on the first (device A) BIG-IP iSeries device:

```
#!/bin/bash
current_device_name=$(tmsh show cm device | grep CentMgmt::Device: | cut -d' ' -f2)
mgmt_device_name=$(tmsh list cm device management-ip | grep management-ip | cut -d'
' -f6 | sed 's/\./_/g')
tmsh mv cm device $current_device_name $mgmt_device_name
tmsh create net vlan HA interfaces add { 1.1 { tagged } } tag 156
tmsh create net self HA address 1.1.1.1/24 vlan HA allow-service default
tmsh modify cm device $mgmt_device_name configsync-ip 1.1.1.1
tmsh modify cm device $mgmt_device_name unicast-address { { ip 1.1.1.1 } { ip
management-ip } }
tmsh modify cm device $mgmt_device_name mirror-ip 1.1.1.1
```

- Run the following script on the second (device B) BIG-IP iSeries device.

```
#!/bin/bash
current_device_name=$(tmsh show cm device | grep CentMgmt::Device: | cut -d' ' -f2)
mgmt_device_name=$(tmsh list cm device management-ip | grep management-ip | cut -d'
' -f6 | sed 's/\./_/g')
tmsh mv cm device $current_device_name $mgmt_device_name
tmsh create net vlan HA interfaces add { 1.1 { tagged } } tag 156
tmsh create net self HA address 1.1.1.2/24 vlan HA allow-service default
tmsh modify cm device $mgmt_device_name configsync-ip 1.1.1.2
tmsh modify cm device $mgmt_device_name unicast-address { { ip 1.1.1.2 } { ip
management-ip } }
```

- Then, run the following script **on only the first** (device A) BIG-IP iSeries device.

```
#!/bin/bash
tmsh modify cm trust-domain Root ca-devices add { 10.105.180.202 } name
10_105_180_202 username admin password admin
tmsh create cm device-group sync_failover_group type sync-failover devices replace-
all-with { 10_105_180_201 10_105_180_202 }
tmsh run cm config-sync to-group sync_failover_group
management-ip } }
tmsh modify cm device $mgmt device name mirror-ip 1.1.1.2
```

- Run the following script **on only the first** (device A) BIG-IP iSeries device to configure a TMM data interface with a non-floating VTEP Self IP address connected to the underlay tenant network.

```
#!/bin/bash
tmsh create net vlan Tenant_183 interfaces add { 1.1 { tagged } } tag 183
tmsh create net self vtep vlan Tenant_183 address 10.105.183.101/24 allow-service
all
```

- Run the following script **on only the second** (device B) BIG-IP iSeries device.

```
#!/bin/bash
tmsh create net vlan Tenant_183 interfaces add { 1.1 { tagged } } tag 183
tmsh create net self vtep vlan Tenant_183 address 10.105.183.102/24 allow-service
all
```

3.7.4. LBaaS Services Installation

To configure the overcloud on the controller, you must SSH from the undercloud node (rh-director-09) to the controller (overcloud-controller-0). Log in as the heat-admin user.

To find the IP Address of the controller node, do the following:

```
$ source /home/stack/stackrc
$ nova list
+-----------------+------------------------+------------------------+
| ID              | Name                   | Networks               |
+-----------------+------------------------+------------------------+
| 978e397e-c96f-… | overcloud-compute-0    | ctlplane=10.105.180.47 |
| acb7f9a8-12ee-… | overcloud-compute-1    | ctlplane=10.105.180.46 |
| 549b8840-e03c-… | overcloud-controller-0 | ctlplane=10.105.180.48 |
+-----------------+------------------------+------------------------+
```

In the (truncated) table above, you will find the IP address that was dynamically assigned to the controller. (Note: These values will most likely be different in your cloud.)

- SSH to the controller and install the F5 LBaaS packages.

```
$ ssh heat-admin@10.105.180.48
$ sudo –i
# cd /usr/src
# curl -O -L https://github.com/F5Networks/f5-icontrol-rest-
python/releases/download/v1.1.0/f5-icontrol-rest-1.1.0-1.el7.noarch.rpm
# curl -O -L https://github.com/F5Networks/f5-common-
python/releases/download/v2.1.0/f5-sdk-2.1.0-1.el7.noarch.rpm
# curl -O -L https://github.com/F5Networks/neutron-
lbaas/releases/download/v9.1.0/f5.tgz
# curl -L -O https://github.com/F5Networks/f5-openstack-
agent/releases/download/v9.2.0/f5-openstack-agent-9.2.0-1.el7.noarch.rpm
# curl -O -L https://github.com/F5Networks/f5-openstack-lbaasv2-
driver/releases/download/v9.2.0/f5-openstack-lbaasv2-driver-9.2.0-1.el7.noarch.rpm
# tar xvf f5.tgz -C /usr/lib/python2.7/site-packages/neutron_lbaas/drivers/
f5/
[root@overcloud-controller-0 src]# rpm -Uvh f5-icontrol-rest-1.1.0-
1.el7.noarch.rpm
Preparing... ################################# [100%]
Updating / installing...
  1:f5-icontrol-rest-1.1.0-1 ################################# [100%]
[root@overcloud-controller-0 src]# rpm -Uvh f5-sdk-2.1.0-1.el7.noarch.rpm
Preparing... ################################# [100%]
Updating / installing...
  1:f5-sdk-2.1.0-1 ################################# [100%]
[root@overcloud-controller-0 src]# rpm -Uvh ./f5-openstack-agent-9.2.0-
1.el7.noarch.rpm
Preparing... ################################# [100%]
Updating / installing...
  1:f5-openstack-agent-9.2.0-1 ################################# [100%]
[root@overcloud-controller-0 src]# rpm -Uvh f5-openstack-lbaasv2-driver-9.2.0-
1.el7.noarch.rpm
Preparing... ################################# [100%]
Updating / installing...
  1:f5-openstack-lbaasv2-driver-9.2.0################################# [100%]
#  curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
#  python get-pip.py
#  pip install ordereddict
```

3.8. F5 OpenStack Agent Configuration

Follow the instructions provided in the F5 OpenStack LBaaSv2 documentation to install and configure the F5 LBaaS components.

F5 OpenStack LBaaSv2 Quick Start Guide

3.8.1 Configure F5 LBaaSv2

Our example */etc/neutron/services/f5/f5-openstack-agent.ini* file is shown below (with the commented lines removed). This VXLAN configuration uses the F5 agent in **L2-adjacent mode**.

```
cat f5-openstack-agent.ini
[DEFAULT]
debug = True
periodic_interval = 10
f5_ha_type = pair
f5_external_physical_mappings = default:1.1:True
f5_vtep_folder = 'Common'
f5_vtep_selfip_name = 'vtep'
advertised_tunnel_types = vxlan
f5_populate_static_arp = False
l2_population = True
f5_global_routed_mode = False
use_namespaces = True
max_namespaces_per_tenant = 1
f5_route_domain_strictness = False
f5_snat_mode = True
f5_snat_addresses_per_subnet = 1
f5_common_external_networks = True
f5_bigip_lbaas_device_driver =
f5_openstack_agent.lbaasv2.drivers.bigip.icontrol_driver.iControlDriver
icontrol_hostname = 10.105.180.201,10.105.180.202
icontrol_username = admin
icontrol_password = admin
f5_parent_ssl_profile = clientssl
```

Remember, we can test both VXLAN- and VLAN-backed networking environments. This example f5-openstack-agent.ini file **will work for both VLAN- and VXLAN-backed** environments. The configuration of your overcloud in the network-environment.yaml file is what controls whether or not neutron will create VLAN- or VXLAN-backed networks.

In other words, even if you define the `f5_vtep_selfip_name` and other VXLAN-related parameters in this file, **all VXLAN-related parameters in this file will essentially be ignored** if you're not using VXLANs in your overcloud. Neutron will never set up VXLAN tunnels for an overcloud that is only configured for VLANs.

3.8.2. Configure Neutron for LBaaSv2

Follow the instructions in the **F5 OpenStack LBaaSv2 Quick Start Guide** to configure F5 LBaaSv2 for your deployment. The online documentation has full descriptions of **all available features** and how to configure each.

Test Services Installation

The testing client is a Docker container instance. The only connectivity requirement to run the validation test is that the container must be able to reach the external API interfaces of the overcloud controller. For this use case, two test suites will be run: the community LBaaSv2 tempest api, ddt, and scenario tests, and a test suite that uses Heat to patch and upload TMOS virtual edition images to the overcloud glance service.

4.1. Install the Testing Client

The testing client must be built on a system that has access to github.com and the Red Hat CentOS repositories. The required software packages are downloaded and installed as part of the build process. Each test environment is populated with the unique software requirements for the specific tests run within the testing environment.

The testing client is intended for interactive use. To that end, the tox system -- which the OpenStack community uses to automate the creation of test virtual environments as part of the CI/CD automation -- is supplanted by installation scripts that are part of the Docker build process.

The testing client also includes several tools that can troubleshoot and query your cloud. The OpenStack cli tools are all installed, which allows you to query your cloud from the test client. The F5 python SDK and iPython are also installed, allowing for interactive orchestration of your F5 BIG-IP devices along with the various OpenStack client APIs.

- To install the testing client package, clone the the repository from GitHub.

```
git clone https://github.com/f5devcentral/f5_openstack_validation_tools
```

- To build the test client Docker image, run the Docker build process.

```
docker build -t f5_openstack_validation_tools ./f5_openstack_validation_tools
```

- To start the test client, run the container.

```
docker run -i -t –name f5_openstack_validation_tools -v
/tmp/bigip_images:/bigip_images f5_openstack_validation_tools
```

Note: Your BIG-IP zip packages (downloaded from downloads.f5.com) must be in the mounted volume (*/tmp/bigip_images* in the example above). As shown in the example, the –v flag mounts a directory from the local machine to the container host. All files accessible at the mounted path on the local machine are also accessible in the container.

Use Case Deployment Testing

5.1. Items to be Tested / Not Tested

We only use community tests to validate that the LBaaSv2 installation is functioning properly. The community Mitaka tempest tests can be installed from the community source github repository. The Tempest tests do not require access to the BIG-IP devices. All functionality is exercised through the Neutron service.

The test package uses the F5-supported Heat template to test TMOS Virtual Edition image patching and installation in the overcloud registry, exercising Nova, Neutron, and Glance services in the overcloud.

5.1.1. Details of Test Scenarios

The community Tempest test suite contains tests which either exercise features unsupported by F5 or that do not run properly outside of the OpenStack CI/CD devstack installation. We blacklist these tests using the Tempest blacklist feature.

The community tests also use hard-coded IP addresses that misuse the IPv4 localhost addresses (127/8) when applied to traffic gateways. These hardcoded addresses cause any gateway implementation that properly declines the use of an 127/8 address as an LBaaS pool member to fail. To avoid these failures, the test client replaces all 127/8 addressese with 128/8 addresses before running the tests. Check Launchpad.net or Red Hat Bugzilla reports for more information about this limitation.

The community tests that are blacklisted are noted in Table 15.

Table 15: Blacklisted Community Tempest Tests

Test	Blacklisting Reason
^neutron_lbaas.tests.tempest.v2.api. test_load_balancers_admin. LoadBalancersTestJSON. test_create_load_balancer_for_another_tenant ^neutron_lbaas.tests.tempest.v2.api. test_load_balancers_admin. LoadBalancersTestJSON. test_create_load_balancer_missing_tenant_id_field_for_admin ^neutron_lbaas.tests.tempest.v2.api. test_load_balancers_admin. LoadBalancersTestJSON. test_create_load_balancer_missing_tenant_id_for_other_tenant	F5 LBaaS does not yet support Neutron Network RBAC
^neutron_lbaas.tests.tempest.v2.scenario. test_listener_basic.TestListenerBasic. test_listener_basic	The F5 gateway sends TCP RSTs for connections sent to a virtual address which has not associated virtual service listening on the specified TCP destination port. The test expects a TCP timeout instead of the explicit TCP RST. The test is in error.

^neutron_lbaas.tests.tempest.v2.scenario. test_session_persistence.TestSessionPersistence. test_session_persistence	Known bug in F5 LBaaS Plugin. https://github.com/F5Networks/f5-openstack-lbaasv2-driver/issues/463
^neutron_lbaas.tests.tempest.v2.ddt. test_health_monitor_admin_state_up	These tests through python TypeErrors when run outside of devstack
^neutron_lbaas.tests.tempest.v2.api. test_pools_admin.TestPools. test_create_pool_for_another_tenant	This test is skipped because of a known Neutron bug.
^neutron_lbaas.tests.tempest.v2.api. test_pools_admin.TestPools. test_create_pool_missing_tenant_id_for_admin	This test is skipped because of a known Neutron bug.
^neutron_lbaas.tests.tempest.v2.api. test_pools_admin.TestPools. test_create_pool_missing_tenant_id_for_other_tenant	This test is skipped because of a known Neutron bug.

5.2. Multi-Tenant Community LBaaS Testing

In your running test client container:

- Copy the overcloudrc file from the directory host and source the file to enable access from the OpenStack clients to your cloud.

```
scp root@10.105.181.30:/home/stack/overcloudrc .. overcloudrc
```

- Run the `init_neutron_validate` test to test your access to your cloud; this verifies whether Neutron has all the required extensions to use the LBaaSv2 services.

```
. init-neutron_validate
```

It should report that all services are present. Neutron reports it supports all the core functionality needed for F5 multi-tenant integrations.

1. Once communications with your cloud are confirmed, initialize the LBaaSv2 validation tests.

```
. init-lbaasv2_mitaka
```

This process communicates with your cloud, querying for required settings and building the appropriate configuration for your cloud. It launches a virtual environment that contains all the required software to run the tests.

2. Once your LBaaSv2 validation test environment is set up, run the tests.

```
./run_tests.sh
```

Running all 301 tests can often take over an hour. The tests create a tempest.xml file containing all the test results in junit format. You can upload that file to any junit reporting tool to show your test results. A summary of your test results is also posted to the console output.

```
======
Totals
======
Ran: 301 tests in 3633.0000 sec.
- Passed: 301
- Skipped: 0
- Expected Fail: 0
- Unexpected Success: 0
- Failed: 0
Sum of execute time for each test: 3164.9254 sec.
```

3. Once the LBaaSv2 tests are complete, exit the LBaaSv2 environment by issuing the 'finished' command.

```
Finished
```

5.3. Single Tenant TMOS Virtual Edition Testing

5.3.1. Tenant TMOS VE Deployment

1. To run the TMOS Virtual Edition patching test, initialize the image importer test environment.

```
. init-image_importer
```

The test starts automatically. In the volume mounted to the /bigip_images directory in your container, it locates and examines the BIG-IP zip packages and, using the F5-supported Heat template, uploads all of your BIG-IP zip packages to the OpenStack Glance service in your cloud.

The test launches two separate Heat templates. The first, which is included with the test client, creates a single Nova server instance with Open SSH and Apache services installed via cloudinit, a tenant network, a tenant router, and a Neutron Floating IP exposing the server to the rest of the testing infrastructure. This instance serves the BIG-IP images to the second Heat template.

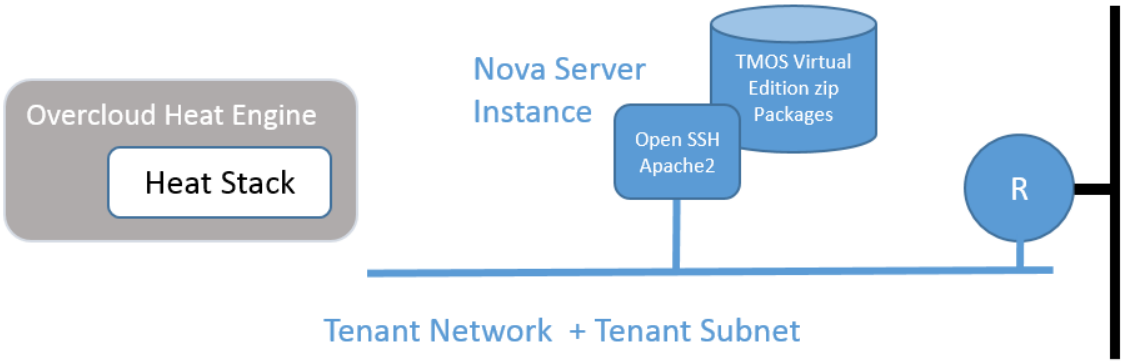

Figure 10: Tenant Network + Tenant Subnet

The second Heat template - the F5-supported Image Patch and Upload template -- prepares the BIG-IP Virtual Edition images for use in OpenStack and uploads them into the Glance registry. A download link for the heat template is provided in the document linked to above.

The image patch and upload template creates a Nova server instance and connects it to the tenant network created by the first Heat template. The Nova server instance uses cloudinit to download the TMOS Virtual Edition zip package specified in the template; extract the appropriate qcow2 disk image; patch the image for OpenStack compatibility; and publish the newly-patched image into the Glance repository.

The template is downloaded, and a Heat stack instance is started, for each BIG-IP image zip package found in the images directory.

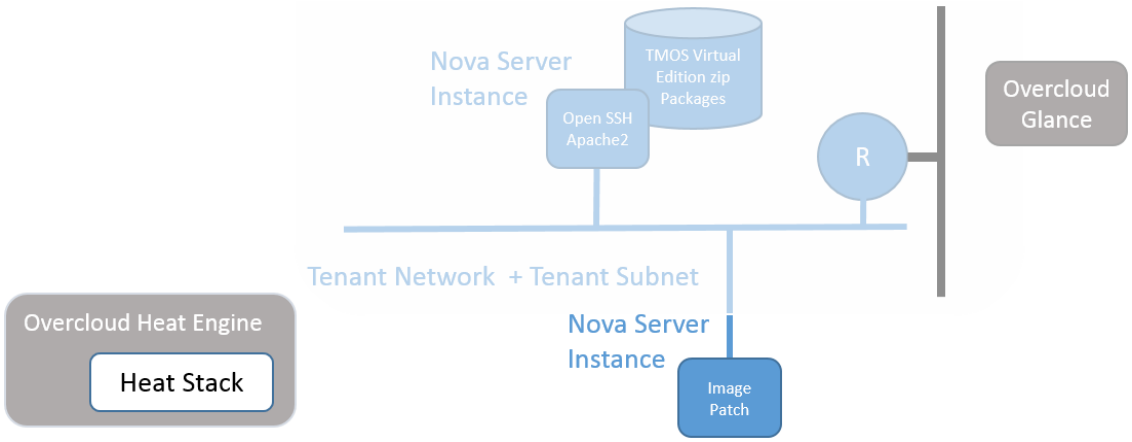

Figure 11: Tenant Network + Tenant Subnet Showing Nova Server Instance

The test suite also extracts and adds any required BIG-IP Virtual Edition Datastor disk images and rename the Glance images in compliance with the TMOS Virtual Edition naming conventions.

2. If completed successfully, the test reports:

```
Images Imported Successfully
```

Performance Testing

Items Under Test / Not Tested

No performance-based tests are being conducted as part of this test plan.